UNIVERSIDAD AUTÓNOMA DE MADRID

ESCUELA POLITÉCNICA SUPERIOR

TRABAJO FIN DE MÁSTER

Integración de redes heterogéneas de próxima generación mediante interfaces abiertas, redes definidas por software e ingeniería dirigida por modelos

Máster Universitario en Ingeniería de Telecomunicación

Autor: SEBASTIÁN LOMBRAÑA, Alberto Juan

Tutor: BRITO MÉNDEZ, Juan Pedro

Ponente: LÓPEZ DE VERGARA MÉNDEZ, Jorge E.

Departamento de Tecnología Electrónica y de las Comunicaciones

FECHA: Junio, 2021

Resumen

En las últimas décadas, los proveedores de servicios de telecomunicación han centrado sus esfuerzos en integrar todos los servicios prestados en redes troncales comunes. Sin embargo, el reto de cursar su tráfico tradicional de fonía, tipos, televisión… con el de los datos que genera el acceso a los sistemas informáticos en Internet ha dado lugar a un escenario heterogéneo que supone aún un desafío para estas compañías.

Los paradigmas actuales de gestión de redes, como el de redes programables —*software defined networking*, SDN—; la virtualización informática de recursos —*network function virtualization*, NFV—; el uso de interfaces abiertas para favorecer la interoperatividad o las técnicas de modelado informático, permiten enfocar estos procesos desde otra perspectiva.

Este trabajo sienta las bases de una línea de investigación con proyección de futuro, ligada a la realización de un programa de doctorado en el que se investigue la integración de nuevos paradigmas en redes de telecomunicación. Por esa razón, tiene se hace un estudio ambicioso de la bibliografía que permita adquirir un conocimiento general sobre el diseño de redes en las últimas décadas y conocer así los porqués de las decisiones que se tomaron.

Con ese razonamiento como base, se diseña, desarrolla y evalúa un un proyecto de integración de redes de telecomunicación usando tecnologías dirigidas por modelos. Este se articula a través de una cadena de producción de código informático que, alimentada con las descripciones de cada una de las fases de la integración, genera los artefactos de código necesarios para el despliegue y evaluación de la red. Esas descripciones son modelos descritos con un nuevo lenguaje de programación de dominio específico, NetAb.

Palabras clave

Gestión de redes, integración de redes, ingeniería dirigida por modelos, redes definidas por software, interfaces abiertas.

Abstract

In recent decades, telecommunication service providers have focused their efforts on integrating all the services provided on common backbones. However, the challenge of handling their traditional telephony, text or television traffic and the data traffic generated by access to computer systems on the Internet has resulted in a heterogeneous scenario that is still a challenge for these companies.

Current network management paradigms, such as software defined networking (SDN); network function virtualization (NFV); the use of open interfaces to promote interoperability and computer modeling techniques, allow these processes to be approached from a different perspective.

This work lays the foundations of a research line with future projection, linked to the realization of a PhD program in which the integration of new paradigms in telecommunication networks will be researched. For this reason, an ambitious study of the bibliography is carried out to acquire a general knowledge about the design of networks in the last decades and to know the whys and wherefores of the decisions that were taken.

With this reasoning as a basis, a telecommunication network integration project is designed, developed and evaluated using model-driven technologies. This is articulated through a computer code production chain that, fed with the descriptions of each of the integration phases, generates the code artifacts necessary for the deployment and evaluation of the network.These descriptions are models described with a new domain-specific programming language, NetAb.

Key words

Network management, network integration, model-driven engineering, software-defined networks, open interfaces.

Índice general

1

Introducción

Son muchas las historias que se aprenden en el proceso de formación, pero rara vez se estudia la Historia. Y esto es así porque el método ingenieril no es metódico. En la ciencia se obra para descubrir y asentar conocimiento, para crearlo, pero también para refutarlo o comprobarlo, y de ahí la necesidad de un método común. En la ingeniería, sin embargo, se hace uso de ese conocimiento, así como de la técnica, la tecnología y la norma, para lidiar con los límites físicos, humanos y económicos de un proyecto. A diferencia de la investigación científica, que sitúa la mirada sobre lo incógnito, la ingeniería centra sus esfuerzos en lo problemático; la primera genera conocimiento y la segunda soluciones. Así, en general, la actividad ingenieril se basa en apriorismos y termina con la firma del proyecto. Ese ejercicio ingenieril no está sistematizado, por lo que *a priori* toma mayor relevancia la experiencia que el experimento: la educativa, la profesional, la vital y las historias. O, dicho de otra forma, en la ingeniería se conocen muchas técnicas mientras se puede carecer de Técnica, al igual que se conocen muchos métodos pero puede llegar a parecer que no dispone de Método.

En las últimas décadas, la evolución de las redes de telecomunicación ha sido abrumador. El ejemplo más claro de ello es Internet, una gran infraestructura heterogénea en la que se entremezclan técnicas, tecnologías y paradigmas nacidos en momentos y contextos distintos. Son muchas las historias que justifican esa evolución. Tanto es así que muchas de las decisiones de diseño de aquellas tecnologías son incluso extrañas a día de hoy, ya que en gran medida se ha perdido el contexto y los condicionantes por los cuales se tomaron. Por ello, podría parecer que se tomaron sin orden ni concierto, sin método. Encontrar una justificación para gran parte de ellas tiene mucha utilidad en el ejercicio de encontrar soluciones adecuadas para nuevos paradigmas.

En otro orden de cosas, la ingeniería dirigida por modelos aspira a proveer de nuevas técnicas y tecnologías útiles para el despliegue, evaluación y operación de esas redes. Esto es así debido que las técnicas informáticas hacen un tratamiento de la abstracción que le son naturales, y en modelado esta característica es evidente. Y si bien no es una metodología que se practique como tal, sí que es posible encontrar muchos de sus postulados en paradigmas como las redes definidas por *software*, abreviadas SDN por el inglés *software defined networking*, o en tecnologías como la familia de protocolos SNMP —Simple Network Management Protocol—. Este trabajo aspira a ofrecer un método de integración de redes basado en esa metodología.

1.1. Motivación

En las últimas décadas, los proveedores de servicios de telecomunicación han centrado sus esfuerzos en integrar todos los servicios prestados en redes troncales comunes, con especial énfasis

en

- agregar el tráfico transportado con tecnologías homogéneas
- mientras que se expone cada servicio provisto con un nivel de abstracción diferente.

Sin embargo, el reto de cursar su tráfico tradicional de fonía, tipos, televisión… con el de los datos que genera el acceso a los sistemas informáticos en Internet ha dado lugar a un escenario heterogéneo que supone aún un desafío para estas compañías. Los paradigmas actuales de gestión de redes, como el de redes programables —*software defined networking*, SDN—; la virtualización informática de recursos —*network function virtualization*, NFV—; el uso de interfaces abiertas para favorecer la interoperatividad o las técnicas de modelado informático, permiten enfocar estos procesos desde otra perspectiva.

De forma adicional, con este trabajo se desea sentar las bases de una línea de investigación con proyección de futuro, ligada a la realización de un programa de doctorado en el que se investigue la integración de nuevos paradigmas en redes de telecomunicación. Por esa razón, tiene especial interés hacer un estudio ambicioso de la bibliografía que permita adquirir un conocimiento general sobre el diseño de redes en las últimas décadas y conocer así los porqués de las decisiones que se tomaron.

1.2. Objetivos

En este Trabajo Fin de Máster se plantea investigar la integración de redes y servicios de telecomunicación mediante técnicas y tecnologías punteras para establecer nuevos procesos útiles para la academia y la industria.

Para lograrlo, en este documento se pretende

- comprender cómo es el actual paradigma de redes de telecomunicación y de sus operaciones de gestión;
- conocer las técnicas y tecnologías emergentes en la operación y gestión de redes de telecomunicación como la virtualización, el programado de recursos de red y el modelado;
- investigar nuevos paradigmas y técnicas de integración de redes y servicios de telecomunicación, así como de su ejecución y evaluación; y
- prototipar un caso concreto de integración de redes heterogéneas fruto de esa investigación, ejecutable y evaluable.

Se trata de unos objetivos amplios debido al carácter marcadamente académico de este trabajo. No se busca el análisis de una técnica, tecnología o proceso concretos, sino abordar los temas que se desarrollan desde un punto de vista formal. En todo caso, dado el carácter profesionalizante del título al que se aspira, se garantiza el desarrollo tangible de un proyecto.

1.3. Metodología

Este trabajo se plantea para una audiencia con cierto carácter multidisciplinar, por lo que se promueve el uso de un punto de vista transversal que involucre discusiones sobre, por ejemplo, la abstracción como herramienta para gestionar la complejidad, la heterogeneidad y la interoperatividad. También, por ese mismo motivo, se exponen algunos conceptos fundamentales que pueden resultar triviales para parte de esa audiencia.

La elaboración de este documento contempla el uso de dos metodologías distintas. Los dos capítulos siguientes son un trabajo de documentación del paradigma actual de las redes de telecomunicación y de algunas técnicas y tecnologías emergentes. Por lo indicado en el párrafo anterior, esa exposición se articula en torno a una serie de conceptos rectores transversales. Por

ejemplo, en vez de relatar los detalles técnicos de una tecnología, se documenta las decisiones de diseño que se tomaron en ella y si tuvieron éxito. Un espacio común en el que poder criticar esos porqués es cómo se gestiona la complejidad en un diseño y cómo la abstracción la pone coto. En concreto, es de interés cómo se consigue la interoperatividad en distintos sistemas, un requisito previo a su integración absoluta.

A continuación, en los últimos capítulos, se propone, diseña, desarrolla y evalúa una solución tecnológica para la integración de dos redes heterogéneas de próxima generación. Para ello, se tiene en cuenta todo lo discutido en los primeros capítulos, para tomar las decisiones de diseño adecuadas al problema. Por último, se hace una discusión crítica de lo expuesto a modo de conclusión.

La información que se recopila procede de fuentes acreditadas, fundamentalmente de libros de bibliografía y de artículos científicos. En los capítulos de análisis, esa bibliografía relata procesos más o menos históricos, discute ciertas decisiones de diseño mediante juicios de valor o explica algunos efectos económicos discutidos o discutibles. En esos casos, siempre se trata de dejar claro qué fuente lo sostiene, incluso a riesgo de resultar repetitivo.

El principal criterio lingüístico es usar el idioma español y su corpus normativo, así como los criterios de léxico y ortotipografía típicos del sector. En general, se prioriza el uso de términos en español incluso cuando son poco usados, por lo que es posible que la elección de algunas palabras resulte extraña; en este documento no hay traducciones propias del autor. Así mismo, se busca evitar el uso de jerga, para favorecer la compresión multidisciplinar. Este documento se ha escrito con el lenguaje de maquetación Latex.

1.4. Fundamentos

Debido al carácter multidisciplinar de este documento se desea detallar algunos conceptos fundamentales para abordar su lectura. En general, son términos conocidos por una audiencia especializada en sus respectivos campos. Sin embargo, dada su especificidad, pueden tanto resultar difusos para parte de la audiencia como incluso ser distintos para dos lectores de distintos ámbitos.

Para no entorpecer la lectura del cuerpo del documento, estos conceptos se han recopilado en el anexo «A Glosario».

1.5. Organización del documento

En el próximo capítulo, «2 Estado del arte. Antecedentes y justificación», se recopila la evolución que han sufrido las redes de telecomunicación e informáticas, con especial atención a las decisiones de diseño que se tomaron en cada momento para gestionar la complejidad mediante la abstracción. Su objetivo es servir de justificación —*rationale*— para la propuesta de integración que se articula después. A continuación, en «3 Estado del arte. Tendencias», se detallan algunas técnicas y tecnologías más actuales y adecuadas para articular los desarrollos de la propuesta.

De esa forma, en en capítulo «4 Diseño» se expone el problema de integración de redes al que se desea dar solución, así como las decisiones de diseño más adecuadas en función de lo documentado antes. En el que le sigue, «5 Desarrollo», se ejecuta el proyecto para articular la solución propuesta, mientras que en «6 Resultados» se comprueba que su funcionamiento es el adecuado con el diseño hecho.

Por último, en el capítulo «7 Conclusiones y trabajo futuro» se detalla lo aprendido en durante este trabajo y las propuestas con la línea de trabajo en la que se desea centrar los desarrollos futuros.

2

Estado del arte. Antecedentes y justificación

Analizar la evolución en el diseño de redes de telecomunicación y de ordenadores en las últimas cinco décadas no es trivial. Sin embargo, en este trabajo se ha tomado la determinación de hacer una exposición trasversal basada en justificar las decisiones de diseño tomadas para la gestión de la complejidad y su interoperatividad. Como consecuencia, no se desea recopilar una evolución técnica y tecnológica a modo de prontuario y, por ello, su clasificación no se puede limitar a una enumeración de cifras, nombres de tramas o programas informáticos. Por el contrario, en un primer lugar, se ha consultado la bibliografía para recopilar esa información técnica con la que, en un segundo lugar, buscar una evolución lógica y coherente.

En este documento se hace referencia únicamente a una serie de redes con las que tanto operadores como prestadores ofrecen acceso al público corporativo y particular de «teleservicios», «servicios digitales», «banda ancha»… No se documentan otras redes, con un carácter más *ad hoc*, como las de radiodifusión o distribución de contenidos, comunicaciones privadas civiles o militares, navegación y posicionamiento, domóticas o comunicaciones náuticas, aeronáuticas o espaciales, por poner ejemplos.

2.1. Taxonomía

El primer punto de encuentro en la bibliografía [1,2] es destacar el esfuerzo, tanto de prestadores de servicios —término que engloba a los actores del ecosistema digital— cómo de operadores de redes tradicionales, de perseguir la convergencia de sus infraestructuras para cursar y procesar tasas de transferencia cada vez mayores desde finales de los 80 y primeros 90 del siglo XX. Y, a diferencia de lo que pueda parecer *a priori*, el reto no se encontraba en dotar de más capacidad al segmento de datos, sino diseñar los mecanismos de control y de gestión compatible y útil para todas las partes.

Antes de detallar ese proceso, la bibliografía coincide en señalar la relevancia del contexto en ese periodo, puesto que influyó en la toma de decisiones técnicas. Debido a su carácter estratégico, las telecomunicaciones están muy reguladas desde sus inicios y mucha de su evolución a respondido a la demandada estatal o militar. Muchos recursos que necesita son de dominio público y sólo disponibles bajo licencia —espacio público radioeléctrico u obras civiles, por ejemplo— y gran parte de su normativa y normalización es desde sus orígenes internacional. Tanto es así que, a nivel global, la competencia en términos de mercado no se permitió hasta casi terminado el siglo pasado. Hasta ese momento, existía en cada país una o varias compañías

prestatarias del servicio en régimen de monopolio —tradicionalmente, distintas corporaciones para la telefonía, la telegrafía y la difusión de radio y televisión—. Durante la desregulación del sector, entre 1984 y 1996, se mantuvieron algunas obligaciones propias del servicio público, aunque se intentó dar un paso hacia adelante y liderar esa prestación integral de servicios digitales. De esa forma, los esfuerzos de los operadores se centraron en desplegar una red única con la que prestar servicios digitales tanto de fonía como de datos, la «red digital de servicios integrados, RDSI», basada en la previsión de una demanda cada vez mayor a partir de la década de 1980.

Según [1], hubo dos estrategias para enfrentarse a ese reto. La primera fue crear nuevos protocolos desde cero basados en agregar y multiplexar más tráfico. La ventaja era inmediata para los operadores de redes de telecomunicación, puesto que se diseñan para ser muy eficientes en su parte troncal. Sin embargo, se constató que esos esfuerzos eran insuficientes para lidiar con parte de la heterogeneidad de las redes —como se discute en «2.2.3 Modo de transferencia asíncrona, ATM»—. La otra estrategia fue dotar al paradigma ideado en las primeras redes de ordenadores, la basada en la pila de protocolos de Internet, de medios más eficientes de encaminado y conmutado para esa parte troncal.

En [2], se explica cómo esos operadores asumieron el desafío de cursar tráfico de distinta naturaleza bajo el paraguas del paradigma RDSI que promovía la convergencia tecnológica de los servicios tradicionales con el acceso a los nuevos sistemas informáticos. Un segundo motivo que señala esa fuente es que apostar por la electrónica de consumo generó una dinámica de mercado en la que, a más demanda satisfecha, mejor era la oferta y más demanda se generaba, debido a la realimentación positiva se aplicar la economía de escala. Así, el efecto revulsivo no fue en que en el momento de desregular el mercado se ofertasen conexiones de unos cientos de baudios mientras que a día de hoy se comercializan de unos cientos de «megas». Es, por el contrario, que en esos momentos un módem costaba cerca de 1000 dólares mientras que hoy esos equipos terminales cuestan unos cuantos dólares. Por último, debido a esa desregulación, señalada como fundamental en el capítulo anterior, surge la obligación de interconexión y arrendamiento de acceso a nuevos prestadores de servicios.

Por lo tanto, en este capítulo se evalúan las dos tendencias principales en el diseño de redes: el enfoque de los operadores de telecomunicación y el de los desarrollos informáticos. El primer enfoque resultó en redes muy potentes pero poco dinámicas. Eran las adecuadas para cursar sus propios servicios, orientadas a la conexión, con un aprovisionamiento de recursos lento —un mismo circuito virtual podía estar dando el mismo servicio durante semanas o meses— y muy fiables —con tiempos de recuperación de 60 ms en las redes ópticas que se documentan en la sección «2.2.4 SDH/SONET»—. Por el contrario, el paradigma de Internet alumbró redes evolutivas y con una gran versatilidad para idear y prestar nuevos servicios muy demandados, sobre todo informáticos. Adolecían, sin embargo, de medios de transporte eficaces cuando las redes escalaban. Para terminar, se documenta el enfoque actual, en el que se tiende a cierto agnosticismo mediante la segregación de las funciones de transporte y las de prestación del servicio.

2.2. Enfoque de los operadores de telecomunicación

Los operadores de redes tradicionales prestaron multitud de servicios sobre una única infraestructura durante décadas, la red telefónica pública conmutada, y se pensó que podría seguir evolucionando usando la ingente cantidad de par de cobre desplegada. Así, se usó en la red de acceso o «última milla» hasta bien empezado el siglo XXI, tramo que se conoce como «bucle de abonado»; si existe tal granularidad es debido tanto al carácter monopolístico de aquellos

operadores como a su mandato público de conexión —servicio universal—. También debido a ese motivo, el acceso a la red debía ser barato, por lo que era una decisión de diseño lógica trasladar parte de la complejidad a la red troncal, en la que se desplegaron medios de transmisión con mayor capacidad, como cable coaxial, radioenlaces y fibras ópticas.

2.2.1. Jerarquía digital plesiócrona

La jerarquía digital plesiócrona, JDP, fue la técnica de transporte propuesta a finales de la década de los 50 para el tráfico de fonía en esos nuevos medios troncales. Se basó en el uso de la multiplexación mediante la división del tiempo para agregar flujos de llamadas, que en esos momentos se digitalizaban con 8000 muestras por segundo de 8 bits. En Europa, se agregaban 32 llamadas en un flujo de 2,48 Mb/s —*E-carrier*, norma para la UIT— mientras que en Estados Unidos, Japón y Canadá se hizo con 24 llamadas a 1,544 Mb/s —*T-carrier*, ideada por la AT&T—. Conforme estos flujos confluían en la red troncal, se iban a su vez agregando, formando flujos de mayor tasa binaria —llamados E1, E2… y T1, T2… —. Estos números tienen relevancia debido a que condicionaron todos los desarrollos posteriores, como se relata en las siguientes secciones.

Las decisiones de diseño fundamentales en esta tecnología son, primero, la de multiplexar en el tiempo, que en décadas anteriores fue mecánica y ahora podía ser electrónica y, segundo, su carácter plesiócrono, «cerca de ser síncrono», que permitía cierto margen de error en la señal de reloj de los equipos terminales —no en fase, si no en velocidad—. Por eso, al agregar las llamadas podían «sobrar» bits que debían adjuntarse al final de las tramas, lo que dificultaba mucho extraer flujos concretos del agregado total sin desmultiplexarlo. Esta decisión asume que parte que parte de la red debe ser menos precisa, fundamentalmente para abaratar los costes en el acceso, pero a cambio el diseño debe ser algo más complejo y añadir en la trama espacio para esos bit extra.

2.2.2. Circuitos virtuales de datos. X.25

A finales de los 60 y primeros 70, surgieron tanto las primeras redes públicas de conmutación de paquetes como servicios de acceso a sistemas informáticos —Tymnet, Telenet en los Estados unidos, CYCLADES en Francia… — orientado a usuarios institucionales y corporativos —se habló de la «teleinformática» [3], la informática remota—. Muchas de ellas bebían de la experiencia que guió los primeros pasos del proyecto ARPANET, pero las decisiones de diseño en las que se basaban eran muy distintas de las que se detallan en la sección 2.3.1. En concreto, y a diferencia de aquella, estas redes se basaron en usar los circuitos virtuales de la red telefónica para el transporte de datos, mediante módems que ocupaban un canal vocal; el diseño asume comunicaciones orientadas a conexión y sin un encaminamiento propio. Los esfuerzos, por ello, se centraron en proporcionar una interfaz común con la que conectar los terminales informáticos con los de telecomunicación. Así, la UIT desarrolló la norma X.25 [4] entre mediados de los 70 y primeros 80, basada en el diseño de esos primeros servicios de transporte de datos, que define cómo se ha de hacer el intercambio de información entre los equipos terminales de datos y los que proveen el circuito virtual.

Lo relevante de esta norma no son sus detalles técnicos, si no el interés creciente por diseñar sistemas capaces de comunicarse entre ellos, incluso cuando todos eran tecnologías propietarias, y la elección de una interfaz física común con la que abstraer los detalles de cada uno.

2.2.3. Modo de transferencia asíncrona, ATM

El modo de transferencia asíncrona, ATM por su nombre en inglés, se ideó como una solución integral para el transporte eficaz del tráfico esperado para la RDSI. Tanto en [2] como en [1] se describe cómo ATM era el candidato de las operadoras de redes para sustituir otras tecnologías que hoy son casi normativas como Ethernet. Su importancia radica en que señaló la necesidad de cursar con calidades de servicio distintas el tráfico de naturaleza heterogénea, lo que supone una gestión de mayor complejidad. Para lidiar con ella, se definieron unos adaptadores con los que segmentar y encapsular la información que se quería transportar: de tasa constante y sensibles al retardo, de tasa variable tanto sensibles como insensibles al retardo y tráfico *best effort*. Así, los detalles del tipo de tráfico se abstraían del funcionamiento de la red y eran los terminales los que debían gestionarla.

Se trata de un mecanismo de transferencia de información orientado a la conexión y con un entramado fijo de 53 octetos denominado «celda», útil para establecer tanto «circuitos virtuales» —necesarios en telefonía— como conmutar paquetes de datos. La elección de una trama tan pequeña era necesaria para reducir la variabilidad en el retardo —*jitter*— en el tráfico sensible a ella como las llamadas telefónicas, pero permitía a la vez hacer un encolado y conmutado de paquetes eficaz. Era una tecnología asíncrona para el usuario, es decir, la red no esperaba que su información llegase de forma coordinada, pero los flujos de celdas que transitaban la red sí que lo eran. En cada trama se identificaba el camino virtual que debía seguir el datagrama y el circuito virtual al que pertenecía, lo que permitía gestionar tanto el tránsito como la entrega de la información en destino. En ATM se ideó la misma conmutación mediante etiquetado que luego prescribiría MPLS, para reservar «canales virtuales» al principio de cada comunicación, y que se expone en la sección «2.3.3 MPLS y GMPLS».

En [5] se detalla cómo ATM no tuvo el éxito esperado debido a que su complejidad era innecesaria en entornos como las redes locales de ordenadores, en las que Ethernet era más barato e igual de eficaz, por lo que los esfuerzos de mercado se centraron en desarrollar las tecnologías de la pila de protocolos de Internet. Se adoptó, en todo caso, los preceptos de calidad de servicio que promovía.

Por su parte, en [6], se hace un estudio más crítico de la evolución y el impacto de ATM. El desarrollo de las tecnologías basadas en esta solución vivió un *boom* en los primeros 90, pero cada actor entendió la tecnología de una forma distinta. Las operadoras entendieron que sería ubicuo, que la gestión de una calidad de servicio extremo a extremo propiciaría la aparición de nuevas prestaciones y que unificaría todos sus servicios en una única red integral multiservicio, incluso en su parte troncal con las tasas prometidas de hasta 100 Gb/s. Por otro lado, la comunidad investigadora predijo el uso de ATM para ofrecer un servicio universal de punto a punto basado en conmutación de paquetes, y se hizo una ingente cantidad de aportaciones para abaratar la tecnología ATM, proponer nuevos escenarios o integrar IP dentro de sus mecanismos.

Esa misma fuente explica, sin embargo, que este tipo de tecnologías necesitan de ciertos efectos de mercado que no se lograron, como el de la externalidad de red por la que la demanda de un actor satisface la utilidad de otro y, por ello, se necesita cierta masa crítica de demanda para que la oferta sea atractiva —es decir, nadie se compra un teléfono si otra persona no se lo compra; es inútil que una sola lo tenga—. Pero, primero, IP comenzaba justo a despuntar con servicios *best effort* que no necesitaban una gestión de la calidad de servicio orientada a la conexión. Y, segundo, muchos de los clientes que optaron por tecnologías de red en los años 90 eran clientes corporativos, no institucionales o particulares influidos por los operadores de red, y en una red empresarial no se necesitaba tampoco este paradigma. Por ello, el mercado de volcó en producir tecnología Ethernet, menos compleja y más barata, y encaminadores IP, debido a

la proyección de su demanda; las ventajas que ofrecía ATM se suplieron en poco tiempo.

Se conluye que ATM tuvo realmente un impacto en tecnologías posteriores como la que se explica en la sección «2.3.3 MPLS y GMPLS», que recogió muchos conceptos como los circuitos virtuales o unificar las funciones de red pero que, a diferencia de ATM, se centró en dar soporte a una región administrativa concreta y no en resolver el problema de extremo a extremo.

2.2.4. SDH/SONET

Tanto la jerarquía digital síncrona —SDH por sus siglas en inglés— como la red óptica síncrona —SONET— ofrecen un servicio de transporte de datagramas sobre medios ópticos. La primera se usa en todo el mundo menos en Estados Unidos y Canadá, donde se usa la segunda norma, pero ambas son equivalentes; se documentan como una sola y, en todo caso, se detallan algunas diferencias. Se idearon a finales de la década de los 80 como la evolución natural de la red de transporte telefónico descrita bajo el epígrafe «2.2.1 Jerarquía digital plesiócrona», que como se ha indicado no podía extraer flujos concretos del agregado total sin desmultiplexarlo. En consecuencia, se especificó una red con equipos síncronos, por lo que se puede identificar con precisión qué bits de un agregado son de un flujo concreto. Pero además, se diseñó para poder transportar el tráfico heterogéneo pensado en el paradigma RDSI en tramas ATM.

Para favorecer la interoperatilidad, lo único que se definió es la trama del datagrama y la multiplexación, que es también una jerarquía de flujos binarios que se van agregando en el mismo tiempo de trama conforme la información confluye en el núcleo de la red. Para conseguirlo, en cada agregado se disminuye el tiempo de bit, para que allí donde se retransmitía uno se retransmitan varios, por lo que el segmento troncal debe ser más preciso y usar anchos de banda mucho mayores. Se puede afirmar que, de nuevo, parte de la complejidad del sistema se delega en la red troncal y se descarga de ella a los terminales y al segmento de acceso. Sin embargo, igual que ATM o JDP, era un mecanismo de transporte muy robusto y orientado a la conexión, pero poco adecuado para de tráfico *best effort* basado en la pila de protocolos de Internet.

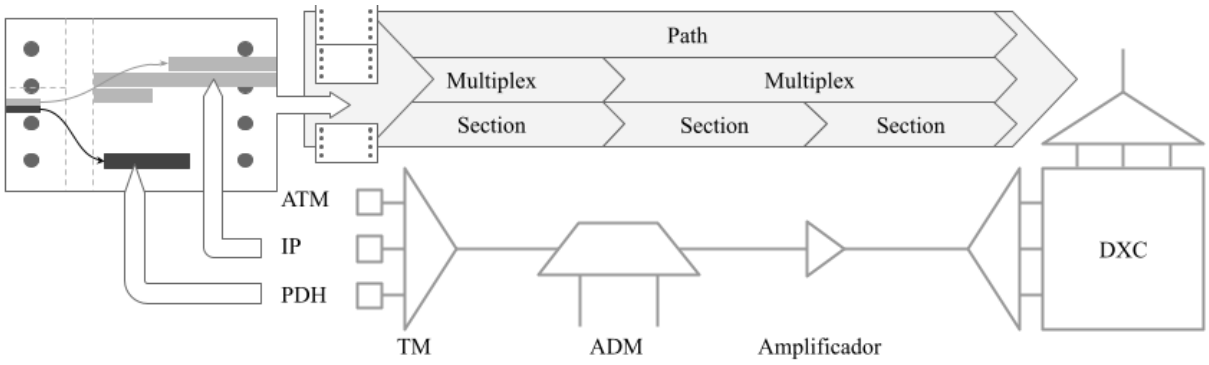


Figura 2.1: Componentes de la red y su uso. Diversos servicios se multiplezan en datagramas y transitan a través de terminales multiplexores —TM—, multiplexores de inserción y extracción —ADM— y transconectores digitales —DXS—. Los datagramas contenían flujos de información con orígenes distintos debidamente referenciados con punteros para poder extraerlos de agregados. Los niveles de abstracción representados se corresponden con los distintos tipos de información que transitaban de forma descentralizada entre los equipos de la red para las funciones de gestión.

Como ejemplifica la Figura 2.1, se compone un servicio de datagramas capaz de transportar tanto paquetes como circuitos virtuales, con tramas de 125 μs de duración. Esto garantiza que

se puedan transmitir las 8000 tramas por segundo que necesitaba una llamada. En en SDH se usa una trama de transferencia básica de 155 Mb/s —flujo STM-1— en tramas de 2430 *bytes*, mientras que en SONET se diseñó una trama menor, de 810 octetos —flujos de 51,84 Mb/s llamados STS-1—; se favorece la interoperatividad igualando la tasa de un STM-1 con tres STS-1. Así, mediante agregados, se prescriben tasas de 155 Mb/s, 622 Mb/s, 1 Gb/s, 5 Gb/s, 10 Gb/s y 40 Gb/s —tráfico nada despreciable en un diseño de los 80—. En cada datagrama se multiplexan datos de múltiples servicios como fonía, datos u otras jerarquías como la JDP o ATM, que se referencian en la cabecera de la trama mediante punteros para poder identificarlos en tránsito; se permiten circuitos virtuales más lentos de 1.5, 2, 3 y 6 Mb/s para cursar servicios de menor tasa.

Se adoptó el modelo de gestión prescrito por la UIT que usa una «red de gestión de las telecomunicaciones, conocido por las siglas del inglés «*telecommunications management network*», TMN, aunque tubo cierta penetración el modelo basado en la familia de protocolos SNMP —Simple Network Management Protocol—»; la explicación de ambos se puede encontrar en la sección «2.5.1 Gestión integrada». Lo más relevante para este caso es que se optó por una gestión y operativa descentralizada, capaz de manejar unos tiempos de recuperación ante fallos en los enlaces de menos de 60 ms. Para lograrlo, con las taras de cada datagrama se podían generar hasta 81 flujos de 64 kb/s para la comunicación entre los equipos de la red, que como era de varios tipos resultaba en cuatro niveles o capas de abstracción. Sobre una capa física se compartía la información necesaria para gestionar la propagación de sección en sección, es decir, entre todos los equipos incluidos hasta los regeneradores de señal más básicos —en SDH y SONET capa de «sección de regeneración» o de «sección»—. Sobre ella, se evalúan las comunicaciones de otra capa para la gestión única de los equipos con funciones de multiplexión —capa de «sección de multiplexado» o de «línea»— que portaba, por ejemplo, la información sobre la precisión del reloj usado para facilitar la sincronía. Por último, la capa de trayecto definía una serie de octetos que permanecían inmutables hasta la entrega de la información y que describían parámetros de la carga útil.

2.3. Enfoque de los prestadores de servicios

En esta sección se expone la evolución de las redes de ordenadores y, en especial, las que se interconectan a la red mundial. Internet es el nombre tanto de la interconexión de redes heterogéneas a nivel global como de los servicios que presta. Para este documento sólo es de interés las decisiones de diseño del germen de esa gran red y, por ello, se analiza la pila de protocolos con los que opera. Su origen es la experiencia en una serie de redes de ordenadores experimentales cuya tecnología divergía de la propuesta por el sector de la telecomunicación. En primer lugar, el objetivo de estas redes era el de interoperar distintos servicios informáticos institucionales de alta capacidad instalados en distintos lugares, por lo que interesaba tener equipos de red lo menos complejos posible. Y, en segundo lugar, el diseño era prácticamente desde cero, sin tener que dar soporte a tráfico de otro tipo o aprovechar la infraestructura previa [7, 8].

2.3.1. Redes de ordenadores

Esas redes experimentales fueron, entre otras, la del instituto metrológico de Reino Unido o las estadounidenses ARPANET y Merit. Se tenía una serie de recursos informáticos a los que se deseaba dar un acceso remoto, por lo que los esfuerzos se centraron en desarrollar los mecanismos necesarios interconectar esos sistemas. La información que se debía cursar era de naturaleza

informática, como credenciales de acceso o resultados numéricos, que es por naturaleza un tráfico de tasa variable e insensible al retardo. Así, se decidió que el mecanismo óptimo para este tipo de transmisiones era la de paquetizar la información y dejar que la red gestionase su tráfico mediante técnicas no deterministas. De esa forma, parte de la complejidad se asume en los terminales y se consigue una distribución más homogénea y descentralizada de la operativa y la gestión.

En una red de circuitos virtuales, las tramas fluyen por un camino prefijado y aprovisionado, mientras que en este tipo de transmisiones los paquetes se memorizan y retransmiten conforme el canal lo permite. Para minimizar los errores en la red, se la descargó de todas las funciones posibles, por lo que muchas de ellas se asumieron los propios equipos informáticos. Como consecuencia, se relajaron los requisitos de fiabilidad de las propias conexiones, con modelos de entrega no orientados a conexión y *best effort*. El resto de funciones de red, como las necesarias para garantizar la conexión o el control de sesión, se ejecuta como rutinas en el segmento informático, de propósito más general, por lo que asumen parte de la complejidad de las comunicaciones. De no ser así, para gestionar la pérdida de un paquete, por ejemplo, los equipos de red tendrían que tener conocimiento de cómo interactuar con cada sistema informático, tener un registro de todos los flujos de paquetes que se le entregan y verificar la recepción con sesiones abiertas en ambos extremos para poder solicitar retransmisiones.

Esto explica porqué este planteamiento se opone al de los operadores de telecomunicación, que buscaban con un segmento de acceso sencillo y barato, que debe ser distribuido a los abonados, y confinar la complejidad de la red en la parte troncal. En este paradigma, por el contrario, los terminales informáticos son por naturaleza muy caros, y es un coste que el usuario o cliente ya ha asumido cuando desea conectarse, por lo que algunas funciones se pueden ejecutar en esos terminales y la red puede ser más barata. Esto favoreció que surgiese un mercado capaz de fabricar equipos de red baratos que, junto con el atractivo de los servicios informáticos como la *web*, tuvieron tanto éxito a partir de los primeros 90.

2.3.2. Ethernet

Entre las muchas tecnologías de transporte adecuadas para el paradigma descrito en el epígrafe anterior, destaca el caso de Ethernet, estándar *de facto* en el despliegue de redes de áreal local corporativas y particulares. Esta norma, en un principio propietaria y luego mantenida por el IEEE,[1] define tanto detalles de la capa física como de la capa de enlace de datos.

Respecto a la primera, su arquitectura se basaba en usar un único bus compartido por varios equipos denominado «segmento». Su evolución ha sido paralela a los requisitos de tasa de transferencia, por lo que existen definiciones tanto para usar cable coaxial, cable trenzado o medios ópticos. De forma adicional, es posible usar equipos para extender los segmentos como repetidores o concentradores, o equipos para mediar entre distintos segmentos como los conmutadores. Por otro lado, las funciones de la capa de enlace de datos se ejecutan en dos abstracciones distintas. En la de control de acceso al medio, la inferior, se gestiona la concurrencia de los sistemas informáticos en cada segmento de red. Así, los ordenadores emiten en el bus de su segmento a discreción, mientras que de forma simultánea escuchan para detectar posibles colisiones y retransmitir —técnica llamada acceso múltiple por detección de portadora, CSMA por sus siglas en inglés—. Mientras que en la capa superior, la del control de enlace, se define la trama usada y el resto de parámetros de la multiplexación.

Además, y es relevante en la evolución del estándar, si en la red existen conmutadores, es en este estrato superior en el que se descubre qué equipos no están en un mismo segmento

[1] `https://www.ieee802.org`

de difusión. De nuevo mediante medios difusivos y escucha pasiva, los conmutadores en la red descubren por qué segmentos a los que prestan servicio se transita para alcanzar cada sistema informático. Se usa para ello el protocolo de resolución de direcciones, ARP, y los propios identificadores de acceso al medio como plan de numeración —identificadores MAC—, para construir tablas de reenvío. En su concepción, es un protocolo muy simple que, de nuevo, descarga mucha de la complejidad de la red en los equipos informáticos. Si, por ejemplo, hay una colisión en un segmento, los encargados de detectarlo y retransmitir son las tarjetas de red de los equipos informáticos, y no otros elementos en la red.

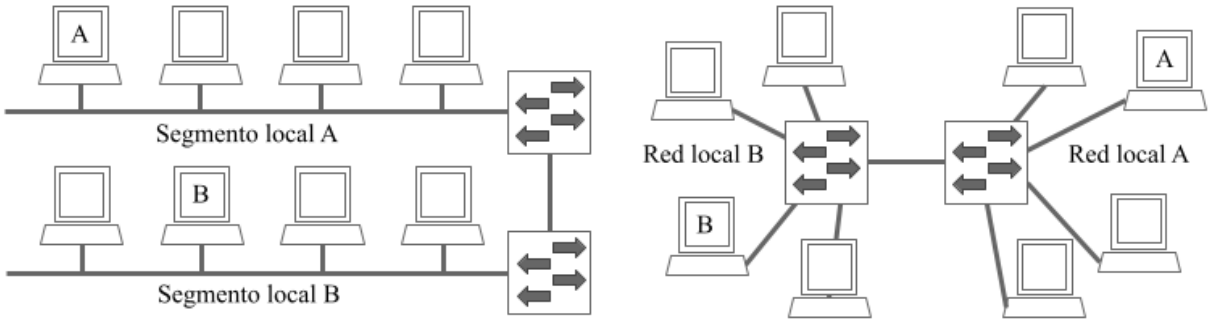


Figura 2.2: Dos redes Ethernet. A la izquierda, la configuración tradicional con un bus compartido; si el equipo A desea comunicar con B, el conmutador de su segmento difunde el mensaje, que acaba llegando a B y respondiendo, de tal forma que la próxima vez los conmutadores ya saben en qué segmento está. A la derecha, la configuración típica actual, con un equipo por bus; aunque el funcionamiento técnico es idéntico, la complejidad de la red es mayor.

En todo caso, conforme la tecnología evolucionó y se abarató, se empezó a usar un único segmento de red para cada equipo informático y la arquitectura de las redes migró de un estilo en bus a uno mallado, con equipos terminales conectados por nodos de conmutación como se ilustra en la Figura 2.2. Se la dotó además de métodos con los que eliminar bucles debidos a la difusión —STP, *spanning tree protocol*—, recuperación y seguridad. Como consecuencia, los equipos de red del estándar Ethernet son ahora tan complejos que pueden asumir toda la funcionalidad básica de la capa de red —la de IP— con su propio plan de numeración y garantías de encaminamiento.

2.3.3. MPLS y GMPLS

La familia de protocolos de MPLS promueve procedimientos únicos para redes heterogéneas y usa definiciones abstractas mucho más cercanas a la ingeniería del *software* que a las redes de telecomunicación clásicas. Fomenta, por ello, la interoperatividad mediante la abstracción, con segregación del plano de datos y el de control, y la redistribución de la complejidad entre las capas clásicas OSI. Para ello, se define por separado los mecanismos de toma de decisiones y los de ejecución, por lo que usa dos planes de numeración, uno para cada plano, y reenvía los paquetes usando etiquetas que se renuevan en cada salto. Las siglas MPLS responden a la conmutación con etiquetas multiprotocolo, muy similar a la que se usaba en ATM.

En [1] se cuenta que esta tecnología fue discutida en el Internet Engineering Task Force, IETF, en la primera mitad de los años 90 para dotar a la pila de protocolos de Internet, TCP/IP, con mecanismos de encaminado y conmutado más eficientes en grandes redes, puesto que Ethernet escala mal al crecer el tamaño de la red. Mediante la conmutación basada en un etiquetado, la complejidad de las funciones de red se redistribuye entre las capas de enlace y

transporte, por lo que ejecuta las funciones de los dos niveles. Además, la carga de las cabeceras posibilita su apilación, es decir, cuando un paquete llega al extremo de otra red se calza otra etiqueta nueva, que es la única que se usa hasta salir de ella, donde se vuelve a usar la primera. Conforme confluyó la tecnología usada en redes de ordenadores locales con la de transporte de los operadores de telecomunicación se trató de generalizar el estilo de reenvío, como MPλS para canales ópticos. Como la ingeniería de tráfico implícita en MPLS trata paquetes, y no celdas, datagramas, circuitos virtuales o canales, fue necesario una familia de generalizaciones llamada GMPLS, Generalized Multiprotocol Label Switching.

La señalización del plano de control en este conmutado permite establecer, mantener, modificar y eliminar caminos con subrutinas que resuelven tanto las operaciones de la capa de enlace de datos como de red. Por ejemplo, cuando un elemento encaminador quiere establecer un camino lo solicita al de destino mediante un objeto informático en el que se informa de lo necesario para ello. Esa información se resuelve, propaga equipo a equipo conforme se establece el camino y se memoriza para ofrecer el servicio. Se permite, en todo caso, que esos elementos de cálculo se ejecuten en remoto y que su resultado se envíe a los equipos, se tal forma que se segregan las funciones de los planos de control y datos —Figura 2.3—.

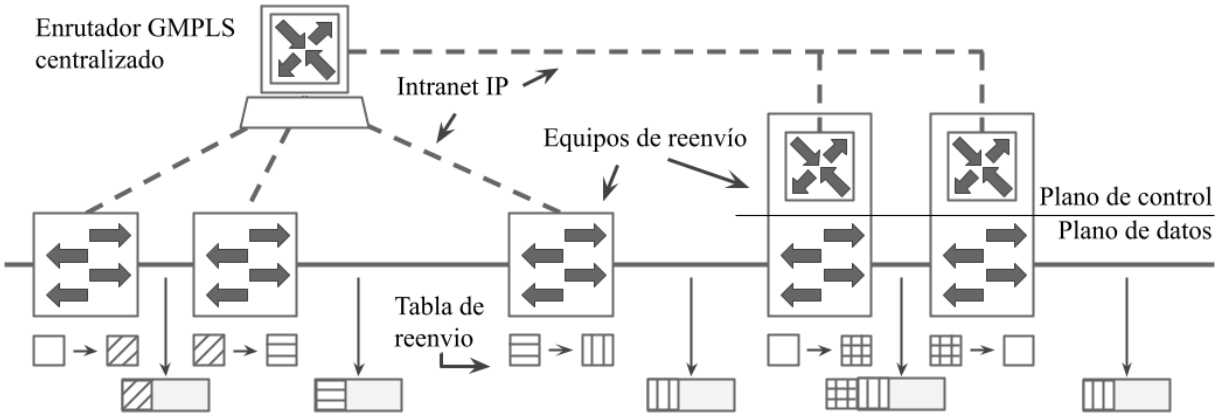


Figura 2.3: Una red GMPLS. A la izquierda, un segmento de red con la función de encaminamiento centralizada, y a la derecha, otro segmento con esa función distribuida. En la parte inferior, las normas de reenvío basadas en etiquetas —y que constituyen un camino concreto calculado bajo demanda—. Obsérvese cómo un paquete ingresa y egresa del segmento de red derecho con la misma etiqueta debido a la técnica del apilado.

Lo más importante de esta señalización, a los efectos del estudio sobre la abstracción y la complejidad que se hace en este documento, es que no es un protocolo concreto, si no una sintaxis abstracta que cada protocolo desarrolla como convenga —con primitivas como *setup*, *confirm*, *downstream error*, *upstream release* o *notify*—. Estos se fundamentan, a su vez, en técnicas de ingeniería del tráfico que identifican los enlaces de la red como recursos abstractos, con información específica como sus capacidades de conmutado —en GMPLS se respeta un orden jerárquico para hacer posible el apilado: MPLS $\rightarrow$ tramas $\rightarrow \lambda \rightarrow$ fibra o puerto—. Además, ofrece interfaces de integración para que los fabricantes provean información administrativa de sus regiones de gestión, tasas de error o calidad de servicio. El resultado es un mecanismo capaz de hacer un integración «vertical» de las distintas tecnologías de conmutación mientras que gestiona una integración «horizontal» entre capas de conmutación análogas en distintas regiones administrativas.

2.3.4. Internet

Conforme creció el número de redes interconectadas se podía empezar a hablar de una red global de comunicaciones [7,8]. Las técnicas y tecnologías que se usan en la internet actual tienen su origen en lo explicado hasta ahora, pero es imposible entenderla sólo con ello. Se desea, sin embargo, dejar una breve reseña de cómo funciona internet.

Internet se compone de un conjunto heterogéneo de redes interconectadas entre sí pero que no comparten los detalles de su implementación; a cada una de estas burbujas de la denomina «sistema autónomo». Y, si bien debido a la convergencia tecnológica las redes que forma Internet son cada vez más homogéneas, tradicionalmente no ha sido así. Por ello, se adoptó el modelo en capas OSI como la norma para favorecer esa interconexión, de tal forma que cada sistema autónomo expone puntos de conexión como una red estratificada; cada ente autónomo no tienen que desarrollar el mismo número de capas, y pueden operar con medios de la primera o segunda capa solamente.

A día de hoy, en Internet cada red compite comercialmente por ofrecer la conexión de usuario a usuario pero cursando el menor tráfico posible. Las redes que proveen de acceso a los usuarios finales deben garantizar su interconexión con otras en los puntos neutros, centros de proceso dedicados a ello —en España, por ejemplo, ESpanix[2]—; de conexión voluntaria pero necesaria para dar servicio a sus abonados. En esos puntos se conectan otras redes que garantizan el transporte internacional y transcontinental. En función del acuerdo comercial que se alcance, el tráfico se tarifa o no; es habitual no tarifar conexiones «entre iguales», pero sí tener que pagar por transitar hacia esas redes de transporte con mayor capacidad de interconexión. De esa forma, cada sistema autónomo compite por inyectar parte de su tráfico en otras redes mientras evita que el resto le inyecte a él, en función de sus intereses. En ocasiones, esos intereses no son claros, y un operador puede tener acceso privilegiado a un servicio muy popular e impedir la interconexión a sus competidores directos obligándoles a transitar por esas redes de transporte troncales.

Por otro lado, destacar que Internet es la infraestructura de las redes de servicios informáticos, como la propia *web*, pero también otros servicios de llamadas, videollamadas, televisión… Se denominan redes *over-the-top*, y pueden ejecutar sus propias funciones de red. Por ejemplo, una red de videojuegos que necesite una sincronización mejor que la que ofrecen los operadores comerciales. Otro ejemplo de redes que usan como infraestructura la internet son las virtuales, aquellas que usan técnicas de tráfico, cifrado… para exponer a nivel aplicación una topología distinta a real.

2.4. Enfoque agnóstico

En la actualidad, los dos modelos descritos conviven de una forma más o menos armónica; el segmento informático despliega todo tipo de servicios a través de Internet mientras que los operadores de servicios de telecomunicaciones facturan a sus clientes tanto por ese tráfico como por otros servicios que serían imposibles prestarlos si los clientes no tuviesen los equipos adecuados —televisores o teléfonos inteligentes, etc.—. Sin embargo, se observa como en el diseño de redes se buscan otro valores. El tráfico cursado evolucionó ha una presencia mayoritaria tipo IP, con tasas de transferencia cada vez mayores y una electrónica al borde de la incapacidad para ejecutar las funciones de la red de forma eficaz. Se impone así el uso de medios ópticos y, en su despliegue, se decide segregar la complejidad de la red de tal forma que se acerque más a la realidad de negocio [2].

[2] `https://www.espanix.net/`

2.4.1. Línea de abonado digital

La línea de abonado digital, DSL, opera sobre los pares de cobre de la red de acceso tradicional para segregar el tráfico de fonía del de datos. Es quizás la tecnología que media entre las aspiraciones del paradigma RDSI con el potencial de Internet a finales de los años 90. Surge en los 80 con el propósito de dar soporte a la prestación del servicio integral, y propone un acceso al medio en frecuencia con el que poder transmitir información por encima de la banda vocal, aprovechando el ancho de banda de los pares de cobre. Sin embargo, su éxito sólo se puede explicar en el marco de la desregularización del sector y las obligaciones de arrendamiento de los actores dominantes. Así, los operadores de telefonía se vieron obligados a ceder espacio en sus instalaciones para que otros proveedores de servicios instalasen sus propios equipos, de tal forma que las tecnologías DSL permitieron el uso del canal por dos operadores distintos, el telefónico y el de interconexión —peajes mediante—. Y, lo que es más importante en este documento, cada operador asume el coste de la complejidad del servicio, y es el nuevo operador el que debe arrendar o disponer de sus propios medios para ofrecerlo.

2.4.2. Red de transporte óptico, OTN

En previsión de sucesivos cambios, una segunda generación de redes ópticas se diseñó para ser transparentes al servicio prestado. Para ello, en el primer lustro de este milenio, se planteó [2] un servicio de transporte cuya oferta no era la de datagramas en los que alojar todo tipo de servicios, si no directamente caminos ópticos —una suerte de λ como servicio—. Para ello se contó con equipos ópticos más sofisticados con los que se buscaba una red más transparente; se habló de una «óptica inteligente» que fuese «*future-proof*» [2]. Se la llamó jerarquía de transporte óptico —*optical transport hierarchy*, OTH— para poder hablar de una evolución de la SDH —a la telefonía se la patentó como «mejoras en telegrafía»—, pero no es una norma jerárquica y el término más usado es el de OTN, por red de transporte óptico en inglés. Es la desplegada en la actualidad.

La idea es que la red proporciona en sus interfaces de acceso un servicio de transporte sobre caminos ópticos que se puede usar para el tránsito de la información, habitualmente de uso exclusivo extremo a extremo —aprovisionamiento de circuitos—. Así, a cada concurrente se le asigna un recurso óptico bajo la premisa que lo debe gestionar como un recurso de la capa física y definir sus propias funciones de red y enlace. En el diseño de estas redes se busca maximizar su transparencia para que los servicios que cursa no se percaten de su existencia. Otro criterio de diseño es potenciar el uso de equipos completamente ópticos, sin intermediación de elementos eléctricos o electrónicos, lo que supone una serie de retos en el tratamiento de señal —como lograr una regeneración con resincronización y reconstrucción, «3R»—, el encolamiento o la diafonía —ecos de la misma señal que se producen por bucles o imperfecciones en los equipos—. Desde un punto de vista teórico, sin embargo, existen dos problemas fundamentales en su diseño que, por sí solos, justifican que no se pueda hablar de una simple capa física: diseñar qué topología puede satisfacer la demanda requerida en cada extremo de la red y, sobre ella, asignar y liberar sus recursos la —Figura 2.4—; son muchas las consideraciones a tener en cuenta, como si existe bidireccionalidad o saltos de canal.

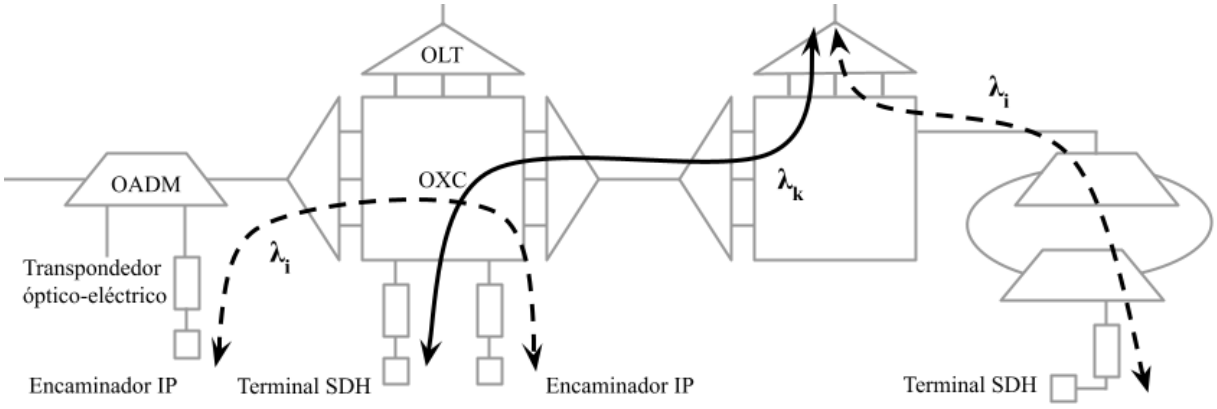


Figura 2.4: Componentes de la red y su uso. Diversos servicios eléctricos se transponden al dominio óptico y transitan a través de terminales de linea optica —OLT—, multiplexores ópticos de inserción y extracción —OADM; ROADM si son teleoperados— y transconectores ópticos —OXC—. Obsérvese que la frecuencia λ_i se aprovisiona para dos transmisiones distintas y condiciona el encaminamiento.

El análisis que se puede hacer de su segmentación en capas o de sus interfaces es similar al que se hizo con la tecnología óptica anterior, y que en este caso se denomina «modulo de transporte óptico» u OTM —Figura 2.5—. Para gestionar el medio óptico y sus equipos se divide la red en secciones entre amplificadores, multiplexadores o interfaces, de tal forma que sus comunicaciones definen 4 capas de abstracción en la gestión de la red:

- OCh, de *optical multiplex section*, que administra el concepto de caminos ópticos asignados a cada usuario de la red;
- OMS, de *optical channel section*, que tiene conocimiento de los tramos en los que existe multiplexación de diversos canales;
- OTS, de *optical transport section*, para la gestión de los tramos entre equipos y su corrección de errores; y
- y la física propiamente dicha —cables, conectores… —.

De forma adicional, para ofrecer transparencia a los servicios que la usan, se definen la adaptación con esos sistemas mediante interfaces que articulan un datagrama «eléctrico» que lleva como tara las cabeceras

- OPU, de *OCh payload unit*, que asocia los canales ópticos con las señales que transporta para gestionar su entrega;
- ODU, de *OCh channel data unit*, que identifica flujos de datos pero desde el punto de vista de la operación, administración y mantenimiento —términos que designa el acrónimo OAM—; y
- OTU, de *OCh channel transport unit*, que identifica cada transmisión entre los equipos de la red —incluidos los regeneradores— para las tareas de corrección de errores, gestión de la calidad de servicio, etc.

Es la información envuelta con la cabecera OTU la que se entrega al canal óptico para su transporte.

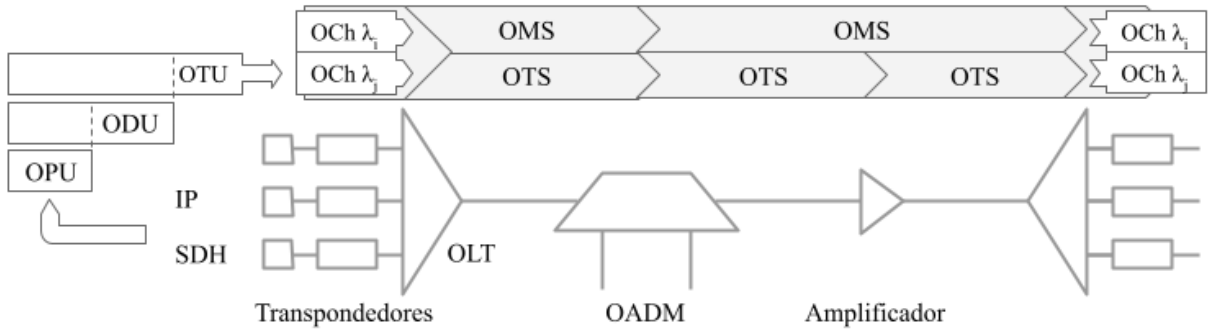


Figura 2.5: La red de transporte óptico usa dos juegos de capas de abstracción. Las comunicaciones entre los equipos usan un sistema de niveles parecido al de SDH, en función de la naturaleza de los equipos que intervengan —OCh, OMS y OTS—. Pero, además, a la información de cada usuario de la red se la carga con cabeceras para gestionar su tránsito y entrega —OPU, ODU y OTU—.

Respecto al reparto de la complejidad que se había hecho con tecnologías previas, cuando al final el grueso del tráfico cursado son datos de la internet, es lógico segregarla de otra forma y, en vez de tener una periferia y un núcleo, se puede diseñar un sustrato de transporte que sostenga la capa de servicios. Así, se usa la misma infraestructura «agnóstica» para cursar tráfico con mucha demanda de calidad de servicio que para transportar tráfico IP; son sus prestadores los que deben hacer su propio diseño. La arquitectura de red que resulta, como se observa en la Figura 2.6, puede ser configurada por cada cliente como considere.

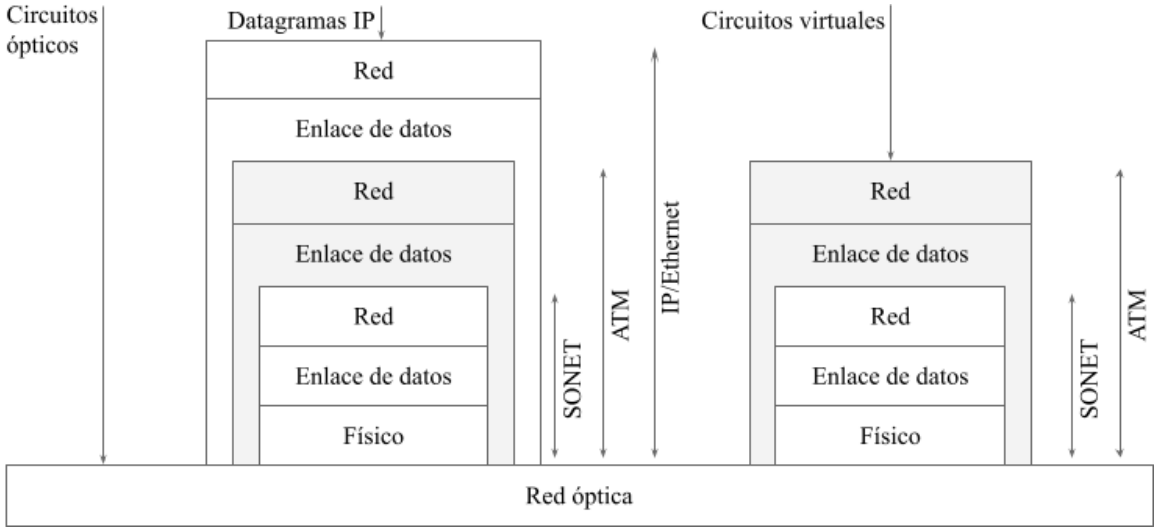


Figura 2.6: Distintas configuraciones para construir servicios sobre la red de transporte óptica. La interoperatividad se logra con más niveles de abstracción que las prescritas por el modelo de capas OSI, en este ejemplo subestratos basados en esa misma norma. Además, aunque no se representan, la red óptica posee sus propias capas.

2.4.3. Red de acceso pasiva

De forma análoga, la tecnología de acceso DSL ha sido sustituida con despliegues de fibra óptica que, de nuevo, buscan maximizar su transparencia y sencillez. En la red de acceso pasiva,

PON por sus siglas en inglés, el medio óptico se despliega sin medios activos desde los terminales de línea, OLT, situados en las instalaciones del operador, hasta los receptores que cada abonado tiene dentro de su domicilio, llamados ONT. Entre esos dos equipos, sólo se interponen elementos pasivos como divisores de haz, con los que las transmisiones de una misma fibra se dividen en varias de ellas para, por ejemplo, dar servicio a un vecindario. Tanto es así que, las transmisiones que la red cursa hacia cada usuario la reciben todos los enganchados a esa misma fibra; se usa cifrado para mantener la privacidad. De nuevo, se usa multiplexación en frecuencia, pero el acceso múltiple desde los abonados hacia la red se gestiona con división temporal. Por último, destacar que existen varias tecnologías y normas que reeditan el término PON muy usados en la actualidad —familia xPON—, como la extensión de Ethernet a estas redes, EPON, o la evolución a velocidades de gigas, GPON.

2.5. Gestión de redes

La «gestión» de una red [9, 10] son las actividades que tienen por objetivo su planificación, organización, supervisión y control conforme a los principios de eficacia, eficiencia y economía. Así, en general, se puede decir que busca encontrar los equilibrios necesarios para que la red preste su servicio con el menor coste posible en recursos.

Con «prestar el servicio» debe entenderse tanto cumplir con las condiciones suscritas con los usuarios como con las distintas obligaciones de servicio universal, interconexión, auxilio judicial... Los «costes» involucrados son, por lo tanto, no sólo los costes de inversión o de explotación habituales en cualquier organización, sino también los relativos a derechos de explotación, obligaciones de paso, obra civil, registro de abonados, planes de numeración o de nombres… Como consecuencia, los costes asociados a un servicio prestado de forma ideal son inviables. El equilibrio entre esos dos grados de libertad se logra concertando una «calidad de servicio» por la que el prestador asume unos deberes a la vez que el usuario suscribe unos límites ideados para modular esos costes —tasa de transferencia, indisponibilidad del sistema, tiempo entre reparaciones… —. La gestión de redes debe articular el compromiso entre la calidad técnica y los criterios de negocio aunque, como se recoge en la Figura 2.7, son también fundamentales las decisiones tomadas en las fases de diseño.

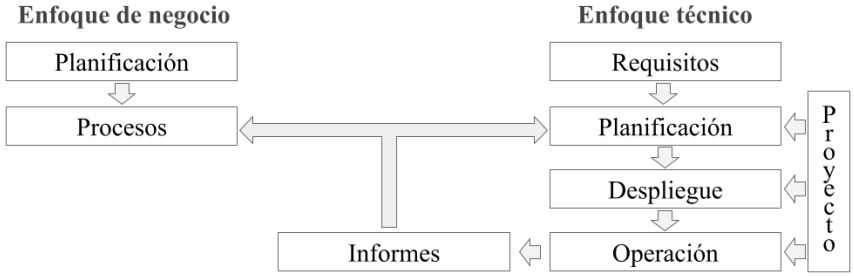


Figura 2.7: Distintas tareas que involucra la gestión de redes y su relación con las áreas técnicas y de negocio.

Los procedimientos que hay que realizar para la gestión de una red difieren en función del marco teórico que adopte la organización, pero está muy extendido el criterio de agruparlos en 5 grupos funcionales denominados FCAPS por las siglas de la gestión de

- fallos, y la supervisión de la red para evitarlos o repararlos;
- configuración, tanto de los equipos de la red como de los sistemas de gestión;

- auditoria y contabilidad de todo lo que ocurra en la red;
- prestación del servicios y su calidad; y
- seguridad, tanto física como virtual.

Esta taxonomía se ideó en un sistemas de gestión integrada promovido por la UIT que se detalla más adelante. Otros sistemas promueven otras clasificaciones y, por ejemplo, el marco ITIL —*information technology infrastructure library*— apuesta más por una clasificación que favorezca la evolución que la propia operación, por lo que promueve clasificarlas entre el diseño, la estrategia, la transición y la operativa. Son funciones concretas, en todo caso y por poner ejemplos, la gestión de inventario, de la topología, de los contratos, de la configuración, de la prevención de fallos, de la facturación, de los riesgos…

Para gestionar una red, el primer paso es definir la información de gestión que se desea recolectar, para después diseñar cómo acceder a ella y cómo procesarla. Hay que tener en cuenta que la información que hay que recolectar tiene naturaleza muy variada, puesto que puede ser estática —como la topología—, dinámica —como la actividad en la red— o estadística —postprocesada—. Para poder conocer la información de los equipos, que en general son de múltiples fabricantes, deben tener el *software* necesario para ello, así como un método común de acceso. En todo caso, se manejan dos paradigmas en este acceso: sondeo periódico o notificaciones ante eventos. El primero descarga de complejidad los equipos, pero la red ha de soportar un tráfico de señalización que puede ser innecesario si la información no ha cambiado. El segundo, minimiza la información en tránsito, pero los equipos han de tener la lógica suficiente para procesar la información y crear las notificaciones. La información puede viajar por la misma red —modelo *mainstream*— o en una red paralela —modelo *sidestream*—.

La heterogeneidad, como la que existe en la internet, tiene un reto adicional para la gestión, puesto que cada técnica, tecnología, fabricante involucrado, etc. impone tanto su propia información de gestión como los mecanismos para su acceso. Por lo tanto, puede ocurrir que sea necesario una cantidad ingente de mecanismos de gestión. Así, el operador de la red debe gestionar distintas interfaces de conexión y distintos programas de gestión, y se arriesga a recolectar información duplicada o a que surjan inconsistencias en ella.

2.5.1. Gestión integrada

Un sistema de gestión integrada [9, 10] trata de realizar todas estas funciones de gestión de la red heterogénea mediante un sistema informático distribuido capaz de recolectar información, mantener las bases de datos, alimentar con ella a las aplicaciones de gestión y mostrar a los operadores unas interfaces de usuario unificadas. Como mínimo, suele ser necesario de una normalización de los protocolos de comunicaciones y de las definiciones sintácticas y semánticas de la información recolectada. Los tres paradigmas de gestión integrada que se van a exponer son el TMN, el de internet basado en SNMP y el usado en redes SDN —aunque este último se explica en la siguiente sección—.

La red de gestión de las telecomunicaciones, más conocido por las siglas TMN, de *telecommunications management network*, es una propuesta que hizo la UIT a principios de este mileno con la que ofrecer un marco común de gestión de redes heterogéneas —tanto analógicas como digitales—. Establecía como requisitos el uso de un protocolo de acceso a la información común basado en notificaciones y que los equipos de distintos fabricantes se expusiesen con unas interfaces con definiciones comunes, como la Q3, que interconectaban distintos dominios de gestión. Este modelo no tuvo éxito, y de él sólo se aprovecho la división de las funciones de gestión de la red FCAPS.

Por el contrario, el paradigma de gestión de internet es el basado en el Protocolo simple de

administración de red, SNMP, del inglés Simple Network Management Protocol. Precisamente por la decisión de descargar a la red de complejidad que se hizo en los primeros momentos de Internet, se ideó un sistema de gestión basado en piezas de *software* distribuidas en cada equipo lo más simples posibles. Como mecanismo de acceso a la información de gestión se diseñó SNMP, con definiciones de información orientadas a objetos descritas con el lenguaje de modelado SMI, adaptación de ANS.1. La configuración de cada «entidad SNMP» se fundamentó en bases de datos de gestión —*management information base*, MIB— a las que se accedía usando SNMP y el identificador del «objeto gestionado» que se desea consultar.

La importancia de este último sistema es, sin embargo, que sienta las bases de una gestión casi total de la abstracción, de cómo cada equipo se expone; es un paradigma dirigido por modelos. La definición sintáctica de la información de cada equipo no se publica en una norma o especificación, si no que es un modelo informático basado en macros ANS.1. Así, el fabricante en cuestión publica esa especificación, y todo agente SNMP puede acceder a ella. Se normalizó, eso sí, una definición concreta de información general, que se la conoce por metonimia como también como MIB y que tenía la capacidad de integrar estas definiciones propietarias. Además, y a diferencia de lo habitual en otros protocolos, este usa definiciones muy cercanas a la ingeniería del *software*; por ejemplo, cada base de datos almacena distintas «instancias» o «ejemplares» del mismo «objeto».

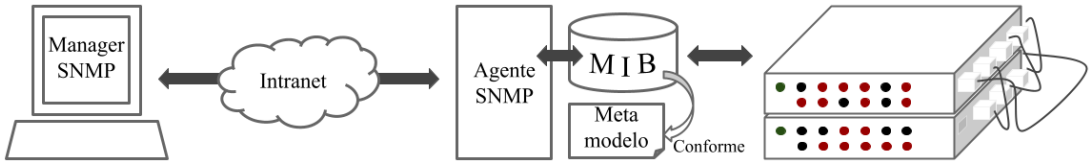


Figura 2.8: En la arquitectura usada en los despliegues de red con el protocolo SNMP, como Internet, se interpone una base de datos, la MIB, que contiene el modelo de información de la configuración de cada equipo de red. Se trata, por lo tanto, de un protocolo orientado por modelos.

2.6. Software defined networking

El paradigma SDN, *software defined networking*, prescribe el uso de técnicas de *software* en el diseño y la gestión que buscan articular redes programables. Este nuevo paradigma se desarrolla en un epígrafe distinto del resto únicamente porque la bibliografía lo hace así; no queda claro que se prescriban decisiones de diseño propiamente dichas más allá de proveer una envoltura informática para favorecer la abstracción y la programabilidad de las funciones de la red.

Este modelo surge en la bibliografía de redes informáticas a finales de la década de los 2000. Hasta entonces, los sistemas se diseñaron con una descripción concreta y, una vez desplegados, permanecían así para sólo evolucionar son su tecnología de década en década; se llamó «osificación» de la red. Este efecto es consecuencia directa de fomentar una economía de escala que fija las interfaces entre los sistemas —como se ha descrito hasta ahora—, puesto que los componentes son más baratos pero menos flexibles y evolutivos. Se propone, así, interponer entre las funciones de red una serie de puntos de acceso programáticos para dotar a la red nuevas abstracciones y, con ellas, de cierta modularidad. La primera premisa, de esa menera, fue desacoplar por completo el plano de control del de datos con una de esas interfaces, por lo que los elementos de reenvío se pueden configurar con definiciones comunes. Se suele prescribir, además, cierta centralización de las funciones de control, como detalla [11].

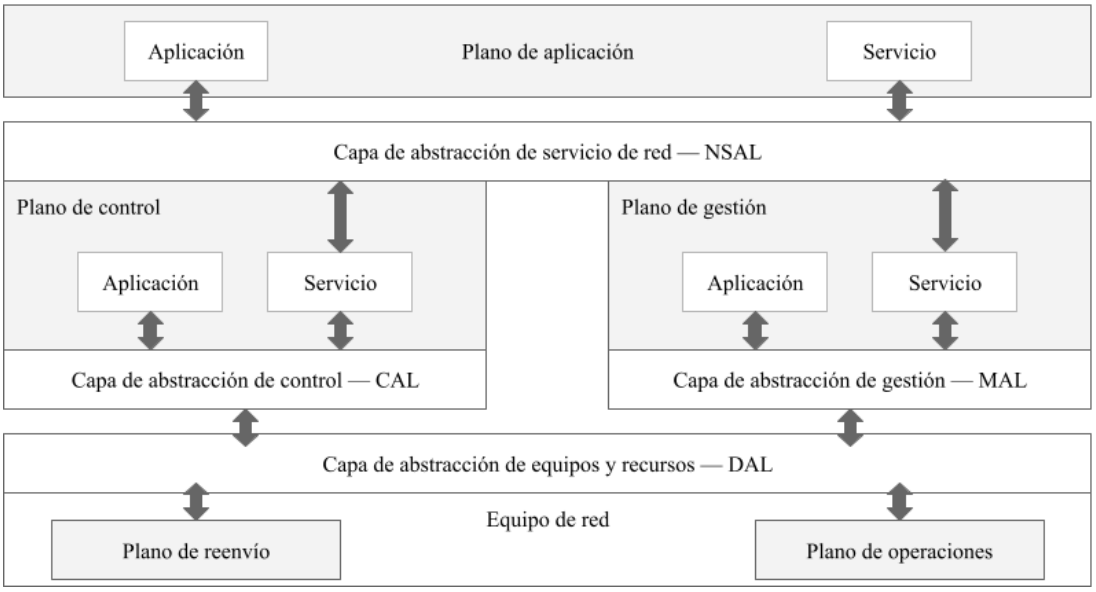


Figura 2.9: Descripción de las interfaces SDN que hace la IETF.

Como consecuencia, se suele describir la arquitectura de red detallando las interfaces dispuestas entre las funciones de la red. Por ejemplo, la arquitectura propuesta en la RFC 7426[3] de la IETF se puede consultar en la Figura 2.9, que describe los distintos planos de la red y sus interfaces. A la división tradicional en planos que se hace en las redes de telecomunicación añade uno de operaciones para la gestión *in situ* de los equipos de reenvío. Todos ellos se conectan mediante las interfaces programáticas descritas, mediante uno u otro protocolo de red, propietarios o libres. Se dota a la red, de esta manera, de una envoltura informática con la que poder gestionarla como si fuesen recursos programables.

Es importante que no se define qué modelos deben exponer las interfaces y, por lo tanto, se abre la puerta a una definición libre de las capas de abstracción entre las funciones de la red —y difundirlas, instalarlas… como cualquier programa informático—. El recurso fundamental que aporta la informática es una gestión natural de la abstracción, más flexible, por lo que no es necesario definir un estándar funcional como el modelo OSI, si no que las interfaces programáticas se definen en cada caso. De esa forma, se hace uso de la versatilidad de técnicas como los modelos de información o las interfaces programáticas para diseñar arquitecturas de red más flexibles y evolutivas. Pero, además, posibilita integrar otras técnicas informáticas de forma casi trivial, como la virtualización de las funciones de la red, NFV.

2.7. Conclusiones

En este capítulo se ha documentado la evolución en el diseño de redes de telecomunicación en las últimas décadas, con el objetivo claro de encontrar un *rationale* o justificación que permita tomar mejores decisiones de diseño. Además, se ha hecho desde un punto de vista transversal, evaluando conceptos como la complejidad y la abstracción. Este trabajo discute la integración de paradigmas de próxima generación en un entorno multidisciplinar y cuyo contexto puede ser muy distinto al estado del arte actual. En consecuencia, conocer esta justificación es fundamental en la discusión y toma de decisiones.

[3] `https://tools.ietf.org/html/rfc7426`

Se ha identificado una tendencia clara en las redes de operadores de telecomunicación para confinar la complejidad de sus redes en la parte troncal, mientras que en las redes informáticas se prefería ejecutar muchas de las funciones de red en el segmento de terminales informáticos. Por último, se ha detallado como el paradigma actual busca gestionar esa complejidad, y sus costes asociados, mediante tecnologías que promuevan la abstracción y la desagregación de las funciones de red, como es el caso de las redes definidas por *software*.

Por último, se ha expuesto la importancia de la gestión de redes, y cómo este sector ha apostado por promover técnicas informáticas para automatizar y favorecer la interoperatividad, sobre todo en escenarios heterogéneos. Estas técnicas son el modelado, la abstracción sintáctica y programática o la virtualización.

3

Estado del arte. Tendencias

En el capítulo anterior se ha concluido que la tecnología óptica es la infraestructura común sobre la que los operadores y prestadores ofrecen sus servicios. Es, además, una apuesta de futuro con la que se pretende tener un segmento de transporte resistente a la obsolescencia. Por otro lado, se ha constatado que muchas de las decisiones de diseño de la red incluyen una capa informática que aprovecha las ventajas ya descritas.

En este capítulo se desea detallar las técnicas y tecnologías más actuales que buscan potenciar esa comunión, con las que poder diseñar una solución que permita integrar dos redes en ese paradigma.

3.1. Programación dirigida por modelos

En este proyecto se asume como premisa la posibilidad de usar ciertas técnicas de la ingeniería del *software* para favorecer un diseño interoperativo que favorezca la integración de redes. No es arriesgado hacerlo debido a que la informática lleva décadas aportando a esta y otras disciplinas técnicas y tecnologías muy adecuadas para automatizar, configurar u operar sus ingenios. Tanto es así que se ha descrito la influencia de las redes informáticas en el diseño de las de los operadores de telecomunicación y cómo incluso han surgido paradigmas sólo posibles con esa comunión, como el de diseño y gestión SDN. Como consecuencia, en este caso se parte con ese sesgo.

La ingeniería dirigida por modelos es un método de trabajo que basa todos sus procesos en estas representaciones. En [12] se explica cómo conforme la técnica avanzaba en las últimas décadas del siglo pasado, surgían nuevas capas de abstracción, sobre todo en sistemas como los informáticos. Se habían diseñado lenguajes de propósito general para no lidiar con la complejidad de programar para cada máquina, surgían nuevas abstracciones como la programación orientada a objetos, así como nuevas herramientas de dominio específico como los programas CAD. Sin embargo, esa tendencia se detiene, y señala cómo cada vez los desarrolladores dedican más tiempo a lidiar con las bibliotecas de código o su mantenimiento que a prestar atención a cuestiones decisivas, como son una visión global que prevenga errores y duplicidades y la debida atención para garantizar la calidad del código y su arquitectura prescrita. Por ello, esa fuente apuesta por el uso de lenguajes de dominio específico para acercar la programación a las definiciones de cada sector y alejarla de la gestión de los recursos de las máquinas.

Así, esta técnica de programación promueve usar los modelos como el principal artefacto en el desarrollo de código. Se dotaría a la ingeniería del *software* de herramientas muy potentes para articular una cadena de producción de código en la que cada nuevo modelo es el resultado de transformar de forma automática uno preexistente. Esto agilizaría el desarrollo, minimizaría

los errores de codificación y facilitaría generalizar los procesos de programación para mantener y distribuir piezas de código compatibles con múltiples arquitecturas o máquinas. Pero además, permitiría, como se ha indicado en la sección anterior, definir nuevos lenguajes con definiciones cercanas a cada uno de los dominios específicos para los cuales se codifica.

Existe un conjunto de normas, fundamentalmente ideados por el Object Management Group, OMG,[1] que definen tanto herramientas como procesos útiles para este tipo de desarrollos. Entre ellos, destaca el lenguaje unificado de modelado, UML por sus siglas en inglés, y el *object constraint language*, OCL. El entorno de desarrollo Eclipse distribuye una implementación de esas herramientas, el Eclipse Modelling Framework, y que es el estándar *de facto* para los desarrollos académicos en esta materia. En ese entorno se pueden ejecutar varias herramientas con las que articular este método de trabajo, como son

- la propia implementación que se hace de UML para la creación de metamodelos, llamada comúnmente Ecore;
- herramientas de visualización de modelos, tanto visuales como gráficas;
- complementos para crear sintaxis concretas, como Xtext para textuales;
- lenguajes de dominio específico para la transformación de modelos como Henshin o ATL; o
- herramientas para sintetizar código como Acceleo.

Existen otras muchas tecnologías de nicho como las descritas en las dos próximas secciones.

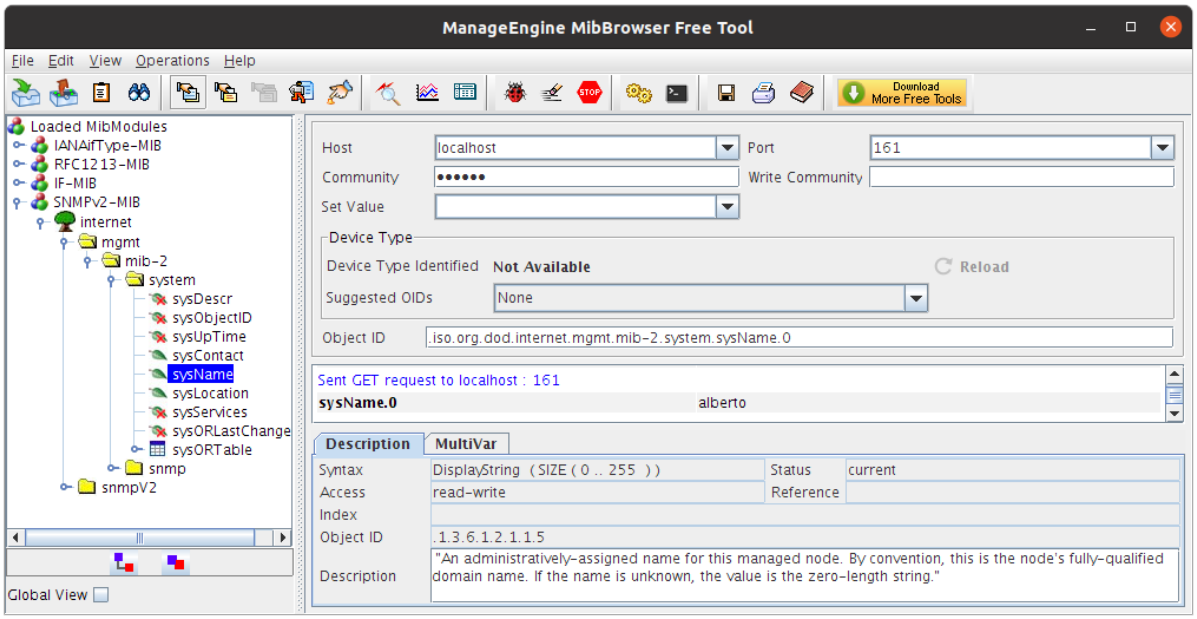
3.2. Gestión de redes dirigida por modelos

Este epígrafe podría perfectamente exponerse como una realidad más que como una tendencia; es, a día de hoy, una apuesta de futuro, pero no es algo novedoso. En [9], un manual centrado en la gestión de redes basada en la familia de protocolos SNMP, y publicado un año antes de que la academia engendrase el término «software-defined networking», ya se dedicaba un par de capítulos a la ingeniería dirigida por modelos; en la Figura 3.1 se muestra una consulta dirigida por modelos de un cliente SNMP. Sea como sea, ahora sí se escribe abiertamente sobre una gestión de redes dirigida por modelos. Conviene señalar que ambos conceptos promueven la programabilidad de las redes tal y como se ha descrito hasta el momento, pero, mientras que el paradigma SDN es una técnica ideada para redefinir el reparto de las funciones de red y su abstracción, este epígrafe describe la búsqueda de todo un sistema de gestión dirigido por modelos.

En [13], por ejemplo, se describe cómo el controlador Opendaylight[2] reúne una serie de técnicas y tecnologías de modelado que prometen lograrlo. Pero, además, explica que se deberían proveer junto a un entorno de desarrollo informático dirigido por modelos y hace referencia a varias técnicas afines: lenguajes de dominio específico, transformación de modelos, generación de código, tecnologías dirigidas por modelos y una cadena común de producción de código con ellas. Otros controladores, como ONOS,[3] también han adoptado esa filosofía —de la Open Networking Foundation—.

Las tecnologías a las que se refiere son, sobre todo, protocolos con un funcionamiento condicionado a la descripción de modelos, fundamentalmente escritos con el lenguaje de modelado YANG [14]. Estos protocolos son, por ejemplo, los que comunican los distintos planos descritos en el paradigma SDN. Esto no es del todo novedoso, puesto que SNMP era capaz de gestionar

[1] `https://www.omg.org/`
[2] `https://www.opendaylight.org/`
[3] `https://opennetworking.org/onos/`



(a) Consulta de un objeto gestionado con el cliente MibBrowser.

```
sysName OBJECT-TYPE
            SYNTAX  DisplayString (SIZE (0..255))
            ACCESS  read-write
            STATUS  mandatory
            DESCRIPTION
                    "An administratively-assigned name for this
                    managed node.  By convention, this is the node's
                    fully-qualified domain name."
            ::= { system 5 }
```

(b) Definición del objeto gestionado consultado.

Figura 3.1: En (a) se observa la consulta de un objeto gestionado con un cliente SNMP. En la parte izquierda, se puede seleccionar el modelo que describe los objetos que se desean consultar, en este caso SYSNAME del modelo estándar SNMPv2-MIB. En la parte central, se observa el ejemplar del objeto que se ha recuperado cuya definición en el modelo se muestra debajo. Esta información es idéntica a la que se muestra en (b), la definición en ANS.1 del modelo consultado.

cualquier equipo siempre que conociese el modelo ANS.1 de su base de datos, la MIB. Por ejemplo, su sustituto, NETCONF [15], funciona de forma muy similar, excepto porque promueve un mecanismo de suscripción y publicación de notificaciones en vez de sondeos.

3.3. Interfaces abiertas

Como se puede consultar en el anexo «A Glosario», una de las técnicas que posibilitan la interoperatividad entre sistemas son las interfaces. Destacar que, en su sentido amplio, una interfaz es cualquier conexión entre dos sistemas. Por ejemplo, ya se ha expuesto cómo, en redes de telecomunicación, es habitual definir como interfaces sus flujos de información —E1, E2…; STM-1, STM-4… —, que no sólo describen su tasa de transferencia sino sus voltajes, la forma de los pulsos o su sincronía. En función del modelo de gobernanza adoptado, estas abstracciones se definen y asientan de muchas maneras: usos, reglamentos, normas industriales, posición dominante en el mercado… Sin embargo, el término «interfaces abiertas» está más cerca de la cultura que surgió en las últimas décadas del siglo pasado alrededor de la informática e Internet.

Muchas de las decisiones de diseño que se tomaron en los desarrollos pioneros de la era digital surgen de la academia, y muchas tecnologías posteriores de usuarios entusiastas como parte de una «comunidad». Por ello, los términos «abierto» o «libre» remiten de alguna forma a aquella cultura, como en «código abierto» o «*software* libre». La industria que supo entender aquella tendencia fue la que pudo posicionarse mejor, y colaboró en aquellos foros en línea con sus intereses. A día de hoy, muchas corporaciones y consorcios industriales participan de esta cultura y publican sus soluciones como proyectos abiertos o libres, y buscan el concurso con aquella comunidad para favorecer su desarrollo e institucionalizarlos como normas *de facto*.

En lo que a este documento se refiere, son dos las técnicas en las que poner el foco: los lenguajes de descripción de interfaces y los modelos de información [16]. Los primeros son lenguajes de dominio específico —y otras tecnologías accesorias como intérpretes o generadores de código— con los que describir la interacción entre sistemas informáticos. Es el caso de, por ejemplo, el lenguaje universal de modelado de la OMG o la descripción de interfaces tipo REST OpenAPI.[4] Con respecto a los segundos, se trata de modelos que especifican con qué parámetros se interactúa o gobierna el sistema, de tal forma que es posible operarlos o configurarlos mediante protocolos o llamadas a procesos remotos. Es decir, no solo describen qué información de gestión que se puede recuperar, si no que mediante su consulta, escritura o borrado se disparan rutinas de gestión con la que configurar los equipos que las subyacen. Un ejemplo que ya se ha expuesto son las descripciones de la información accesible con el protocolo SNMP, las MIB. Otros ejemplos de estos modelos son la descripción Transport API [17] —T-API, de la Open Networking Foundation—, que describe una capa de abstracción con la que gestionar redes de transporte en arquitecturas SDN; Openconfig[5] [16], para operar equipos de red; o la interfaz Openroadmn[6], que aspira a configurar equipos ópticos como los descritos en el capítulo anterior.

3.4. Redes ópticas definidas por software

En el caso de las redes ópticas, también existen esfuerzos para dotarlas de cierto nivel de programabilidad. En [18] se motiva mediante ejemplos, algunos más abstractos que otros, y pone

[4] `https://www.openapis.org/`
[5] `https://www.openconfig.net/`
[6] `http://openroadm.org/`

de relieve la heterogeneidad de sus tecnologías para concluir que existe la necesidad de disponer de tecnologías capaces de simplificar la gestión de la red. Admite, en todo caso, que una solución integral basada en este paradigma necesitaría igual de componentes programáticos con los que atacar cada función o tecnología distinta. En otro ejemplo más concreto destaca que no son pocas las ocasiones en las que los operadores y prestadores necesitan integrar nuevo *hardware* y que, por ello, tienen a equipos de programadores informáticos; las soluciones SDN facilitarían estas situaciones.

En esa misma fuente se recopila qué tecnologías ópticas han sido integradas en el paradigma de redes programables. Destaca el esfuerzo notable

- en segregar los planos de control y de datos en los muchos componentes del substrato fotónico, aunque expone la falta de estudio de escenarios realistas con un enfoque integral;
- en conseguir un control integral tanto de elementos preparados para este paradigma como de otros —como GMPLS—, aunque se desconoce la conexión que existen entre los recursos dedicados a ese control y el desempeño en escenarios realistas;
- en virtualizar recursos, incluso en redes de acceso pasivas o híbridas; y
- en analizar el desempeño de la red con aplicaciones de alto nivel, sobre todo en cómo prestar la calidad de servicio adecuada y reaccionar ante sus fallos, aunque de nuevo señala la falta de estudios en escenarios amplios.

Se señala, en todo caso, que han existido pocos esfuerzos aún en tareas de orquestación de alto nivel, como la que se hace entre distintos dominios de encaminado o conmutado. Analiza también qué retos existen en la escalabilidad, en la evaluación del desempeño, en la integración multinivel o multidominio; indicar, por último, que destaca el reto en la privacidad, seguridad y fiabilidad que supone un control centralizado de las funciones de la red —y asegura que el modelo SDN se fundamenta en esa centralización—.

Son muchísimas las tecnologías orientadas a este paradigma, tanto aquellas puramente SDN como de otras técnicas afines. Respecto a las primeras, destacan todas aquellas interfaces abiertas expuestas en la sección anterior, como Openconfig, Openroadm o Transport-api, junto con sus tecnologías de acceso como el protocolo NETCONF [15]. Y, por supuesto, los controladores centralizados como el Opendaylight u ONOS —aquellos dirigidos por modelos mencionados hace dos secciones—. Con respecto a las segunda, son de especial relevancia algunas funciones de red virtualizadas, como el conmutador Openvswitch —operado con el protocolo Openflow— y el simulador de redes Mininet.

De hecho, una de las motivaciones de este trabajo era analizar un caso de uso alojado por la ONF[7] que demuestra la integración de una tecnología basada en paquetes en la red óptica usando el paradigma SDN. Para ello, se hace uso del controlador que apadrina esa fundación, ONOS, y un desarrollo de la red virtualizada Mininet con encaminadores ópticos.[8] Como consecuencia, es necesario justificar que no se hayan podido usar esos recursos en este proyecto: ha resultado que las máquinas virtuales que se proveían como tutorial ya no son accesibles,[9] y aunque se documenta cómo configurar el entorno desde cero tampoco se ha logrado debido a que se usaban piezas de código desfasadas.[10].

[7] `https://wiki.onosproject.org/display/ONOS/Packet+Optical+Convergence`
[8] `https://github.com/vmehmeri/mininet-optical-wifi`
[9] `https://wiki.onosproject.org/display/ONOS15/Download+packages+and+tutorial+VMs`
[10] `https://wiki.onosproject.org/display/ONOS/The+Packet+Optical+Dev+Environment`

3.5. Redes ópticas de paquetes

En [19] se expone cómo conseguir un aprovisionamiento con paquetes sigue siendo una prioridad en aquellas redes de transporte que cursan mayoritariamente tráfico tipo IP. Es de entender, por ello, que existe una tendencia a conseguir un encaminamiento de paquetes en redes ópticas o, al menos, un diseño que se comporte como tal. El objetivo es tener definiciones de bajo nivel capaces de operar lo más cerca posible del segmento óptico. Eso supone asumir parte de las funciones de red imprescindibles para los operadores, como la señalización de gestión y de la calidad del servicio o métodos de recuperación ante fallos.

En [20] se señala como candidatos a las especificaciones Carrier Ethernet y MPLS-TP, de *transport profiles* y también señalada por la primera fuente, [19]. La primera norma extiende las capacidades de Ethernet para agregar su tráfico conforme transita hacia el segmento troncal, mientras que la segunda proporciona un servicio de pseudo-circuitos. MPLS-TP usa el mismo encaminamiento mediante etiquetado pero presta un servicio orientado a la conexión y simétrico, para lo que incorpora ese tráfico de gestión y consigue tiempos de recuperación menores de 50 ms, en el orden de SDH y SONET. En todo caso, esta tecnología descansa sobre la red óptica en mayor o menor medida; en el segmento troncal es especialmente eficiente, mientras que en el de acceso los prestadores pueden querer tener más control. Esto permite, por ejemplo, que el prestador del servicio clasifique el tráfico según su calidad de servicio en distintas redes virtuales cuyo tráfico es así entregado a la red de transporte según su interés. Por último, [20] también describe extensiones de Ethernet para transportar las jerarquía síncronas y sus señales de reloj, por lo que se postulan como un sustituto adecuado de las tecnologías descrita hasta ahora.

3.6. Tecnologías cuánticas

Los ingenios capaces de explotar propiedades cuánticas son otro de los grandes temas en boga. Para este trabajo, es de interés cómo unas «comunicaciones cuánticas» pueden dar soporte a estas tecnologías. Para abordar este epígrafe se usa esta recopilación, [21], que señala como principales campos a los que dar servicio la metrología —destaca la radioastronomía—, la criptografía y la informática distribuida —como la computación «a ciegas»—. La disciplina, por su parte, se puede beneficiar de una sincronización y de una generación de números aleatorios mejores. Por mercadotecnia, a este tipo de comunicaciones se las suele denominar «internet cuántico».

En una telecomunicación cuántica se cursa información cuántica, aunque suene redundante, un tipo de información [21] basada en la fenomenología microscópica. Esta debe cumplir ciertas condiciones de linealidad y conservación de la energía, por lo que estos mensajes no se pueden inspeccionar, duplicar ni reconstruir y son muy sensible tanto al ruido. Es, sin embargo, de mucho interés debido a que, mientras que en el equivalente tradicional el recurso fundamental es el símbolo —bit, trit, nat… o baudio—, se describen más recursos que el símbolo cuántico —cúbit, cútrit, cúdit… —. Entre ellos destaca el «entrelazamiento», un recurso —en tanto que debe existir de forma previa a su uso— que permite ejecutar operaciones con efectos locales pero a distancias considerables; existe toda una teoría para describir estos recursos [22].

Este tipo de información se genera en experimentos electromagnéticos y fotónicos, nucleares e iónicos, superconductores…; se usa, así, tecnología de microondas y fotónica, resonancias magnéticas y espintrónica, o crioelectrónica, por poner ejemplos. En todos esos casos, el resultado es una entidad física cuya descripción más acertada es la de la mecánica cuántica. Una tecnología que es de especial interés es la fotónica, puesto que sería capaz de usar la red óptica desplegada en la actualidad. Habría, eso sí, que despojarla de elementos activos y no lineales, como amplificadores, rectificadores o conversores de frecuencia. Se usan con éxito despliegues pasivos

```
259     # Network
260     alice_node = Node("Alice", port_names=[ALICE_QUANTUM_PORT, ALICE_CLASSICAL_PORT])
261     bob_node = Node("Bob", port_names=[BOB_QUANTUM_PORT, BOB_CLASSICAL_PORT])
262     quantum_channel = QuantumChannel("quantum_channel_A_B",
263                                      length=fibre_length,
264                                      models={
265                                          "delay_model": FibreDelayModel(),
266                                          "loss_model": FibreLossModel(
267                                              p_loss_length=prob_loss_length,
268                                              p_loss_init=prob_loss_init)
269                                      },
270                                      depolar_rate=0)
```

Figura 3.2: Captura de una prueba que se ha hecho con Netsquid; como se puede observar, usa primitivas muy cercanas a la definición de redes como NODO o CANAL.

como los descritos en la sección «2.4.3 Red de acceso pasiva» para criptografía. Es el caso de redes pioneras como la de Viena [23] o la de Tokio [24], de otras más grandes como la china [25], o incluso SDN como es la de Madrid [26]. En todo caso, son muchos los esfuerzos dedicados a obtener mecanismos de corrección de errores eficaces, pero sigue siendo un problema no resuelto y el principal cuello de botella de estas tecnologías.

Dado su actual interés, existen muchas herramientas disponibles capaces de aprovecharse de la fenomenología cuántica. Muchas de ellas surgen de aplicaciones y experimentos concretos, mientras que otros buscan ser de propósito más general. Son conocidas las herramientas que, por ejemplo, tiene la IBM para uso abierto, la biblioteca en Python Qiskit[11] y sus procesadores cuánticos; otros conocidos no son de uso abierto. Sí lo son algunos de los recursos de Rigetti[12] o Dwave,[13] esta última pionera desde 1999. De la experiencia académica, por poner un ejemplo, surgió Netsquid,[14] con primitivas muy cercanas a la telemática como «nodo» o «canal» —Figura 3.2—, pero orientado a un paradigma de distribución de entrelazamiento —el llamado «internet cuántico»—.

3.7. Nube, IA y otras tendencias

En la bibliografía consultada en las secciones anteriores se exponen también propuestas para integrar otras técnicas propias de la informática en una red con interfaces programáticas, de forma análoga a la virtualización de funciones de la red —NFV—. Muchas de ellas tienen como promotores los propios fabricantes de equipos, mientras que otras son muy estudiadas por la academia. Son, en todo caso, una evolución natural de la flexibilidad que proporciona el paradigma SDN como, por ejemplo, ejecutar las funciones de red de alto nivel a entornos en la nube, usar la inteligecia artificial en su análisis o encadenar las funciones de red usando la modularidad de los desarrollos informáticos.

Por último, existen numerosos esfuerzos en integrar las tecnologías definidas por *software* en entornos en producción, o viceversa; son las redes «híbridas». Por ejemplo, algunos controladores SDN incorporan una implementación del clásico SNMP para estos propósitos —SNMP4SDN[15]— .

[11] https://qiskit.org/
[12] https://www.rigetti.com/quantum-computing/
[13] https://www.dwavesys.com/
[14] https://netsquid.org/
[15] https://docs.opendaylight.org/projects/snmp4sdn

3.8. Conclusiones

En este capítulo se han recopilado algunas técnicas y tecnologías punteras con las que diseñar una solución de integración de redes.

Se ha constatado cómo existe un gran interés por las tecnologías dirigidas por modelos en la gestión de redes de telecomunicación e informáticas. Como consecuencia, es posible encontrar muchas tecnologías de nicho que usan esa metodología, como el controlador para redes SDN Opendaylight o el protocolo de acceso a configuraciones NETCONF.

Además, se ha comprobado lo extendido que está el uso de interfaces de comunicación normalizadas que, además, se distribuyen en forma de modelos. Muchas de estas especificaciones se publican además con un carácter «abierto» para favorecer la interoperatividad. Son, en todo caso, una técnica muy adecuada para promover técnicas de abstracción.

Por último, se ha identificado a la integración de tecnologías cuánticas en redes como una tendencia asentada. Su potencial, así como el interés que suscitan, las convierten en un candidato ideal para un proyecto de integración como el que se desea hacer.

4

Diseño

En este capítulo se determina un proyecto de integración de redes que tenga en consideración lo documentado hasta ahora. En primer lugar, se detalla el escenario al que se quiere dar solución, mientras que a continuación se propone un diseño para ello.

4.1. Descripción del escenario

Se idea un escenario compuesto de dos redes de transporte, una óptica y otra cuántica, que han de integrarse bajo un mismo paradigma de operación y gestión. Ambas redes ofrecen como servicio el transporte entre dos puntos, pero la infraestructura óptica tiene más nodos intermedios, como se observa en la Figura 4.1. La red óptica tiene capacidad para operar a nivel de red, pero no así la cuántica, que sólo proporciona un servicio de enlace experimental. Por último, la red óptica opera con un paradigma de red definido por *software*, por lo que posee sus características típicas como una operación segregada del plano de control y del de reenvío.

Este escenario se ha escogido así porque son muchos los esfuerzos actuales en integrar estos paradigmas, como se pudo comprobar en la sección «3.6 Tecnologías cuánticas».

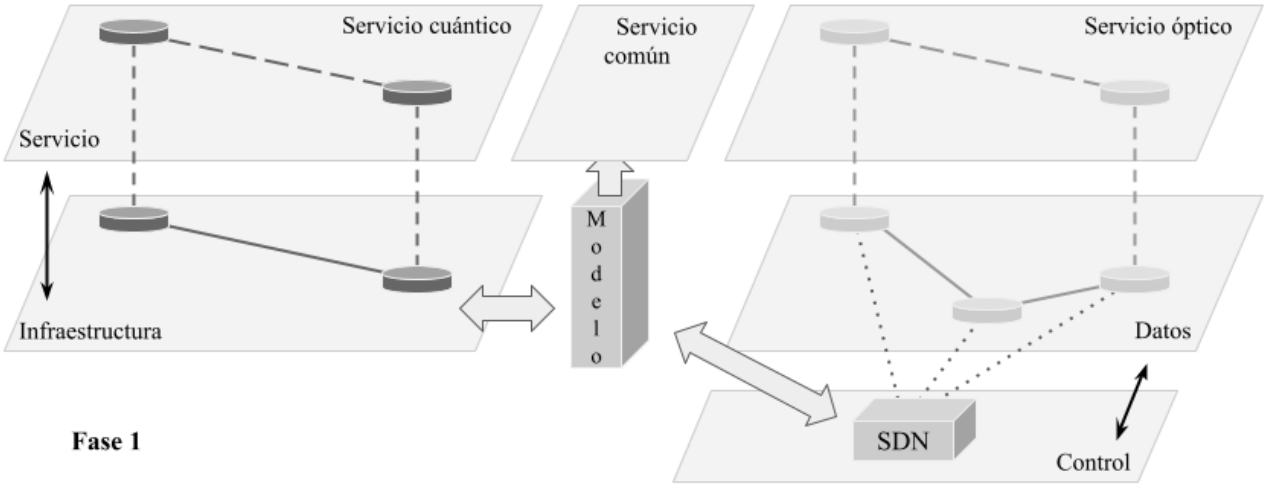
La integración se ha propuesto en tres fases, que deben ser evaluadas en los desarrollos:

1. una primera, que evalúa las dos redes por separado bajo una abstracción común que sirva de guía y evaluación del resto de fases;
2. una segunda, que reúne la gestión de ambas en el mismo controlador; y
3. una tercera, que hace uso de una infraestructura común.

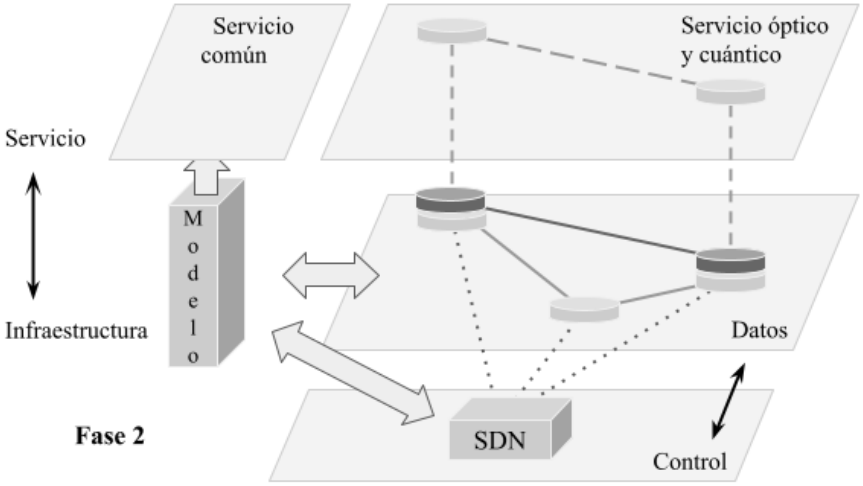
Este planteamiento asume la hipótesis de que al exponer las dos redes como una sola bajo una misma capa de abstracción, esta debe permanecer inmutable conforme el proyecto transite por sus fases. Puede, por ello, servir para evaluar el progreso de la integración. Como se ha discutido en la sección «2.6 Software defined networking», al dotar a las funciones de red de una envoltura informática se logran dos efectos:

- en primer lugar, abstraer los distintos planos, segmentos… de la red con una envoltura informática que medie entre ellos; y
- en segundo lugar, controlar qué información y qué puntos de acceso a subrutinas se exponen.

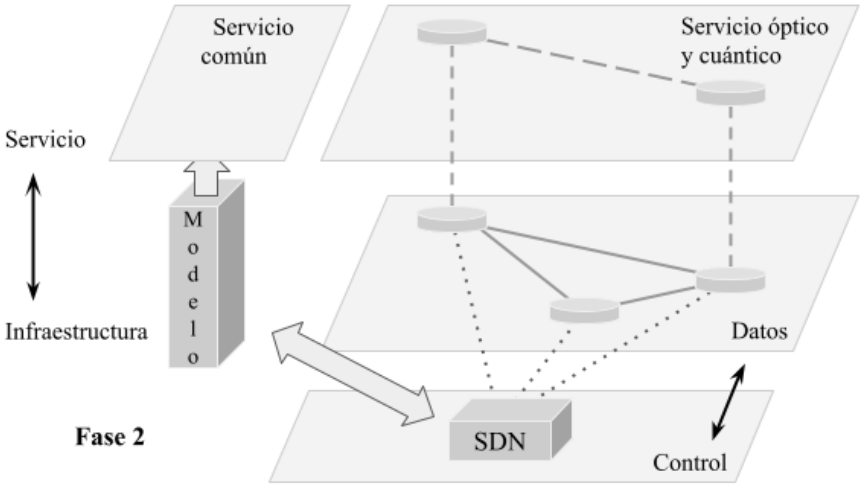
Como consecuencia, debe ser posible evaluar el servicio prestado por una red si se esconden sus detalles de bajo nivel bajo una capa de abstracción y, así, evaluar la evolución de la integración.



(a) Fase 1 del escenario.



(b) Fase 2 del escenario.



(c) Fase 3 del escenario.

Figura 4.1: Fases del escenario propuesto de integración. El procedimiento dirigido por modelos que se debe proponer ha de modelar cada una de estas fases para, alimentado con esas descripciones, generar los recursos adecuados para el despliegue y evaluación de cada fase. En la primera fase, (a), las dos redes operan por separado y exponen sus capacidades en un modelo de información común. En la segunda fase, (b), se integra la gestión de ambas, pero sus infraestructuras siguen estando segregadas. Por último, en la tercera fase, (c), la integración es total.

4.2. Descripción de la solución

A continuación se exponen las principales decisiones de diseño que se han tomado para articular una propuesta que dé solución al escenario descrito.

4.2.1. Detalle de las técnicas y tecnologías

El proceso de integración se articula con tecnologías dirigidas por modelos, una apuesta clara en este trabajo. La documentación ha constatado que es una herramienta adecuada para la gestión de la complejidad en el diseño de redes de telecomunicación. En primer lugar, en la disciplina ya se usan algunos de sus preceptos, como los modelos de información en la gestión SDN o basada en SNMP. Y, en segundo lugar, se ha expuesto lo esencial que es en los diseños una gestión de la abstracción como la que provee el modelado informático.

Dado que se ha formulado como hipótesis principal para dar solución al problema de integración la posibilidad de ocultar las fases de la integración bajo una capa de abstracción común, se procede a elegir un modelo adecuado para ello. Y puesto que el escenario se compone de redes de transporte, se elige como abstracción común la Transport API de la ONF, que define un modelo de información muy amplio para el control de este tipo de despliegues. A pesar de su nombre, Transport API es un modelo de información, no una interfaz programática al uso; en el paradigmas de gestión basados en protocolos de configuración como SNMP o NETCONF —SDN— se accede a bases de datos conforme a estos modelos para alimentar y ejecutar las rutinas de código adecuadas. Por su especial dificultad, este diseño no prescribe desarrollar esas piezas de código, sino únicamente una representación de la red basada en ese modelo.

Para usar el resto de técnicas de este método de programación, se propone modelar un lenguaje con el diseño de redes como dominio semántico específico. Este no debe centrarse en ningún paradigma, técnica o tecnología concretos. De esa forma, se debería poder describir con él cualquier escenario heterogéneo. Se promueve, por lo tanto, un diseño generalista que consiste en recopilar los conceptos con los de se han descrito las distintas tecnologías expuestas en el capítulo «2 Estado del arte. Antecedentes y justificación» y se evaluar sus relaciones.

El modelo de la sintaxis abstracta de ese lenguaje será el metamodelo con el que alimentar el resto de tecnologías dirigidas por modelos disponibles en el Eclipse Modelling Framework. Con las definiciones del lenguaje, es posible crear modelos que representen cualquier tipo de red y persistirlos en memoria en formato XML —realmente, un subconjunto llamado XMI—. Y, con ellos, articular una cadena de producción de artefactos de código con los que simular, operar y evaluar las redes en integración. Esto se consigue usando transformaciones de modelos que acerquen la descripción de alto nivel a los detalles de cada artefacto, basadas en la herramienta Henshin, así como generadores de código capaces de inspeccionar los modelos, como hace Acceleo. Además, existe la posibilidad de dotar al lenguaje de una sintaxis concreta ajena a la básica que provee Eclipse, y se elige usar Xtext para crear una gramática textual. En todo caso, con otras herramientas se podría tener sintaxis concretas gráficas o de realidad aumentada, por ejemplo; la sintaxis abstracta sería la misma.

La tecnología que se propone para la simulación o emulación de la red óptica son el controlador ONOS y la red virtualizada Mininet. El primero tiene una operación mucho más sencilla que otras alternativas, mientras que la segunda ofrece en un único programa todo lo necesario para esa simulación. Sin embargo, como se ha justificado en «3.4 Redes ópticas definidas por software», usar la extensión de Mininet óptica se ha evaluado como inviable, por lo que se renuncia a emular de forma adecuada este tipo de redes —*e. g.* conmutación de circuitos—. En cuanto a la red cuántica, se propone usar la biblioteca Qiskit de la IBM, escrita en Python y que

que permite usar procesadores reales; su desarrollo se comienza desde cero. Se han descartado otras opciones, como Netsquid porque está orientado a simulaciones académicas y no permite ejecuciones distribuidas. Para alimentar al modelo Trasport API con el que se ha de abstraer el conjunto, se pueden desarrollar sendos conectores tipo REST con la tecnología OpenAPI. Los experimentos se realizarían en una máquina virtual Debian 10 sobre un Ubuntu 20.04.

Para terminar, a modo de evaluación, se propone generar y comparar los modelos de información Transport API, tanto los esperados como los generados, en cada una de las fases de la integración. Esto debe permitir ratificar o refutar la hipótesis que se ha formulado sobre la conveniencia de usar una capa de abstracción para evaluar cómo evoluciona la integración.

También se proyecta la generación de pruebas sencillas que comprueben la viabilidad y estabilidad de la solución propuesta. A modo de ejemplo, se propone analizar la conectividad mediante el envío de paquetes ICMP —programa Ping—. Con esto, se pretende poner en relieve esa posibilidad de generar tantos artefactos de código como sean necesarios para la simulación, operación e integración de la red. Sin embargo, en este trabajo no es de interés recolectar y analizar métricas del rendimiento de la red, ya que lo que se discute es el procedimiento dirigido por modelos y su utilidad.

4.2.2. Detalle de las fases

Para la primera fase, se proyecta diseñar la subred óptica con Mininet compuesta de tres nodos y tres enlaces y controlada por ONOS; es un ejercicio trivial. Por el contrario, la subred cuántica sólo constaría de un único enlace codificado con Qiskit.

En las sucesivas fases, las dos redes deben operar de forma conjunta aunque tengan una tecnología incompatible. Como la red cuántica es un único enlace que no tiene equipos de red *per se*, la tarea se reduce a proporcionarle un medio de acceso desde los extremos de la red virtualizada. Por la forma de virtualizar redes que tiene Mininet, los programas que se ejecutan en sus nodos virtualizados tipo clientes no tienen acceso a Internet, pero Qiskit necesita conectar con las máquinas remotas de la IBM. Para permitirlo, se pueden configurar a los equipos de reenvío de la Mininet para que sirvan de pasarela a la máquina anfitriona.

De esta manera, se idean dos aplicaciones cliente para ser desplegadas en esos extremos de la red virtualizada y con la capacidad de iniciar la simulación, informar al otro extremo de este evento, y que aquel obtenga los resultados. Este diseño, que se puede consultar en la Figura 4.2, delega toda la complejidad de la red en su parte troncal, esa ejecución basada en Qiskit, y sólo delega en el extremo de la red esas aplicaciones terminales con las que iniciar la comunicación.

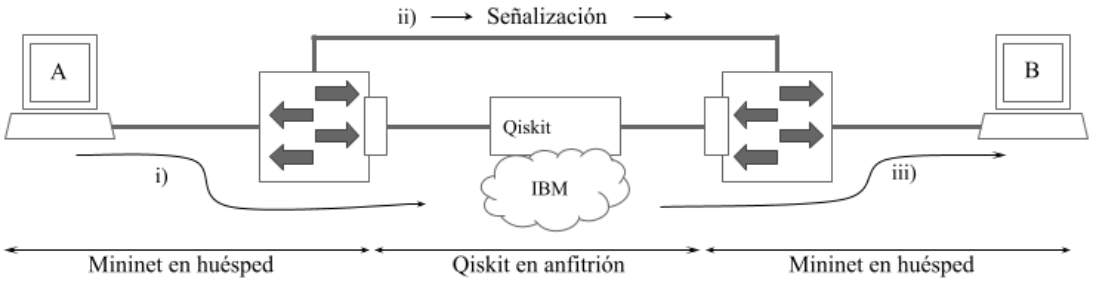


Figura 4.2: Escenario con el que simular la red cuántica. i) El cliente A solicita a Qiskit —en la máquina anfitrión— la emulación, y ii) informa a B para que iii) acceda a por el resultado por su interfaz.

Se escogen las direcciones 10.0.1.0/24 como el plan de numeración IP en las interfaces de la

subred óptica mientras que el subespacio 10.0.2.0/24 se dedica al servicio cuántico. Esta decisión debe permitir tener un control efectivo del tráfico cursado cuando se ejecuten pruebas sobre el conjunto. En la segunda fase, los equipos de reenvío de las dos subredes serán distintos, pues sólo se integra la gestión, mientras que en la tercera fase los dos servicios deben funcionar en la misma infraestructura de red.

4.2.3. Detalle del flujo de trabajo

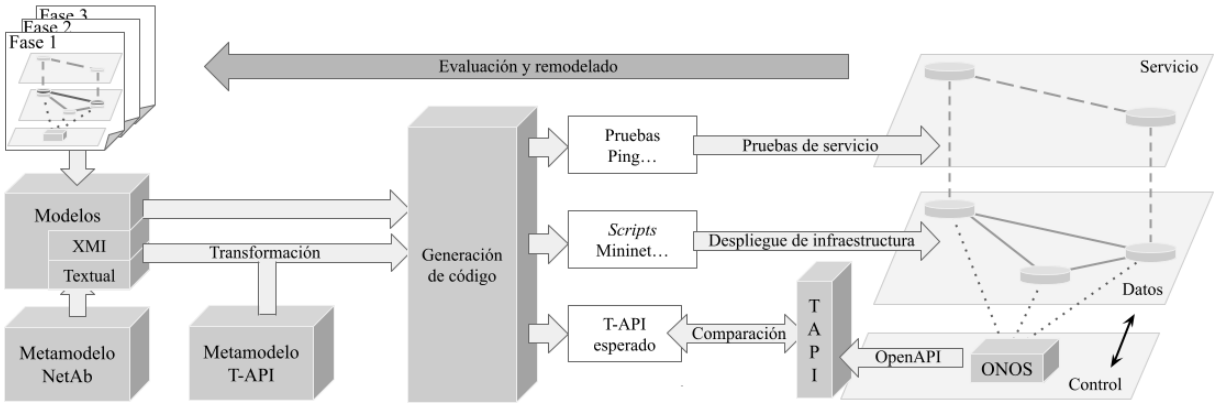


Figura 4.3: Flujo de trabajo propuesto en este proyecto para la integración de redes heterogéneas con medios dirigidos por modelos.

A modo de conclusión, se puede presentar todo el diseño como un flujo de trabajo que se asemeja a una cadena de producción de código informático como las que se prometían en la sección «3.1 Programación dirigida por modelos». Y es que, si se aprovecha la programabilidad de las redes que usan el paradigma SDN, así como otras técnicas informáticas como la virtualización de funciones de red —NFV— es posible simular, despliegar, evaluar y operar una red de telecomunicación como si se tratase un proyecto de *software*. Esta idea no es nueva, y es habitual usar la NFV en centros de proceso de datos para crear y destruir recursos con los que adaptar el sistema a la demanda, por ejemplo. Sin embargo, no es tan habitual usar herramientas dirigidas por modelos que faciliten o automaticen la creación de los artefactos de código necesarios para ello.

4.3. Conclusiones

En este capítulo se ha descrito un escenario concreto en el que proceder al ejercicio de integración. Se trata de dos tecnologías heterogéneas y de paradigmas distintos: una red óptica con gestión SDN y un enlace experimental de naturaleza cuántica.

Se ha formulado una hipótesis según la cuál es posible controlar la evolución de la integración mediante una capa de abstracción que recopile información de las redes que la subyacen. De esa forma, conforme la integración transita entre sus fases o pasos, el modelo de información recopilado debe ser coherente con el esperado.

Se ha concretado que esas fases son *i*) evaluar un funcionamiento separado; *ii*) evaluar una gestión conjunta y *iii*) evaluar una gestión y funcionamiento conjuntos. Además, se ha presentado el flujo de trabajo dirigido por modelos que representa el proceso de modelado, generación de código, despliegue y evaluación de la red y su integración.

Se ha elegido usar la tecnología de virtualización Mininet y el controlador ONOS para la red óptica, mientras que se escoge usar Qiskit para la simulación cuántica. Como modelo de información, se ha escogido usar la interfaz abierta Transport API, que se publica además como una especificación OpenAPI con la que generar artefactos de código.

5

Desarrollo

En este capítulo se exponen los desarrollos hechos para ejecutar el proyecto propuesto.

5.1. Lenguaje de dominio específico

Se compone el modelo del lenguaje propuesto, que se puede encontrar en la Figura 5.1, con la implementación del lenguaje universal de modelado que tiene el entorno de Eclipse; este modelo se denomina «metamodelo» y representa la sintaxsis del lenguaje creado.

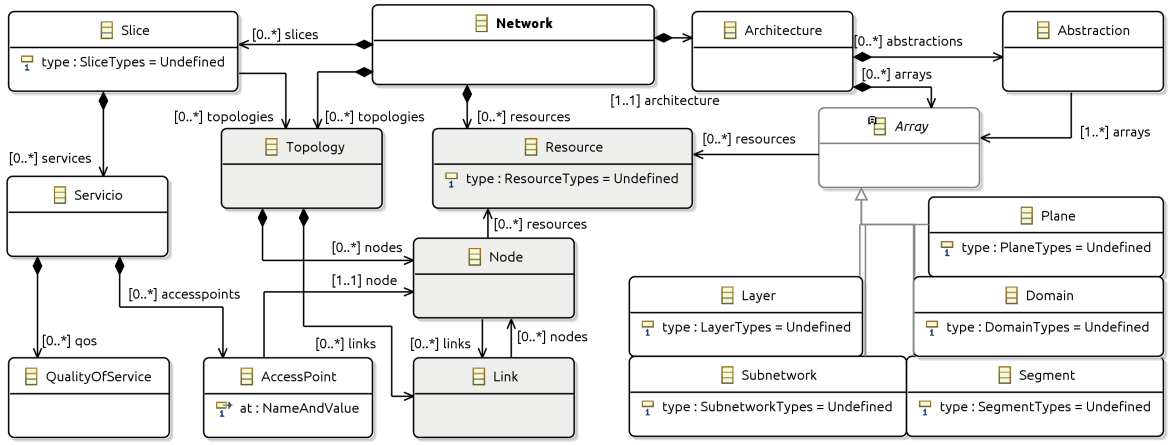


Figura 5.1: Sintaxis del lenguaje propuesto, «NetAb», y sus módulos: en el centro y gris, «NetSign»; a la izquierda, «NetServis», y a la derecha «NetArc». Se recomienda comenzar su lectura desde la clase NETWORK, arriba, y detenerse en cada módulo. Se han eliminado los atributos NAME y UUID para facilitar la lectura, y no se muestra el tipado usado para la arquitectura —términos como «plano de gestión», «segmento de acceso», «capa de transporte» o «dominio de reenvío»—.

El lenguaje se ha llamado NetAb, ya que puede describir de forma abstracta cualquier red. Se especifica cualquier red como una serie de funciones —recursos— a lo largo de su topología; se modela con el módulo que se ha llamado NetSign. Por otro lado, con esas estructuras se prestan servicios, que además necesitan las definiciones de calidad de servicio o puntos de acceso descritos por el módulo NetServis. Por último, NetArc permite definir la arquitectura de la red, esto es, qué abstracciones se desean interponer entre qué capas, segmentos, dominios, planos o

subredes. El diseño es tal que todas estas definiciones heredan de un único término, que agrupa una serie de funciones de red; para este lenguaje, todas esas definiciones son sinónimos. Esta modularidad permite diseñar la red con la topología y funciones de red necesarias, que suelen ser requisitos, para ser después cuando se gestiona la complejidad de repartirla entre distintas capas o planos o construir servicios sobre ella.

Se desarrolla una gramática textual con ayuda de la herramienta Xtext. Esta herramienta, también integrada en el EMF, hace uso del metamodelo para generar todos los artefactos de código necesarios para poder editar el lenguaje, así como un DSL propio para concretar la gramática de forma casi inmediata. En «B Ejemplo de red programado con NetAb» se puede encontrar una muestra de la sintaxis, mientras que en la Figura 5.2 se recogen algunas sentencias junto con parte de la definición de su gramática.

NetAb.xtext

```
1  grammar master.tfm.v2.NetAb with org.eclipse.xtext.common.Terminals hidden()
2
3  import "http://www.eclipse.org/emf/2002/Ecore" as ecore
4  import "platform:/resource/master.tfm.v2.ecore/model/netab.ecore"
5  import "platform:/resource/master.tfm.v2.ecore/model/netab.ecore#//netsign" as netsign
6  import "platform:/resource/master.tfm.v2.ecore/model/netab.ecore#//netservis" as netservis
7  import "platform:/resource/master.tfm.v2.ecore/model/netab.ecore#//netarc" as netarc
8
9  Network returns Network:
10     ('@'descriptions+=NameAndValue EOL ( '@'descriptions+=NameAndValue)* EOL )?
11     'Network' name=EString EOL
12     TAB 'uuid' (EOL TAB2)? uuid=EString EOL
13     (TAB 'resources' EOL TAB2 resources+=Resource ( EOL TAB2 resources+=Resource  )* )?
14     (TAB 'slices' EOL TAB2 slices+=Slice ( EOL TAB2 slices+=Slice )* )?
15     (TAB 'topologies' EOL TAB2 topologies+=Topology ( EOL TAB2 topologies+=Topology )* )?
16     EOL TAB architecture=Architecture
17     EOL;
```

(a) Muestra de la gramática del lenguaje.

```
Network Fase 3
  resources
    Resource[Device] equipo_de_red
  slices
    Slice[Service] servicio_integr
  topologies
    Topology infraestructura
      nodes
        Node equipo_unico
          resources equipo_de_red
  Architecture despliegue_integral
    arrays
      Plane[Control] control
        resources equipo_de_red
    abstractions
      Abstraction unica_abs
        arrays control
```

(b) Uso del lenguaje.

Figura 5.2: Sintaxis concreta del lenguaje propuesto.

5.2. Fase 1

En esta sección se recogen los desarrollos que se han hecho como parte de la primera fase de la integración. Puesto que en esta etapa las dos redes permanecen aún completamete segregadas, el objetivo es concretar los códigos informáticos que representan a las dos redes, ensayar su síntesis con el lenguaje ideado y obtener esa capa de abstracción común basada en Transport API.

5.2.1. Subred óptica

Como se ha justificado en el capítulo anterior, se usa el programa Mininet para la virtualización de este segmento de red. La forma más versátil de usarlo es proveyéndole un código escrito en Python con el que atacar su API. Ese código es muy académico, por lo que usa primitivas como «nodo», «enlace» o «interfaz» que facilitan la generación de código. De esa manera, se configura la topología descrita en la Figura 5.3 y se despliega.

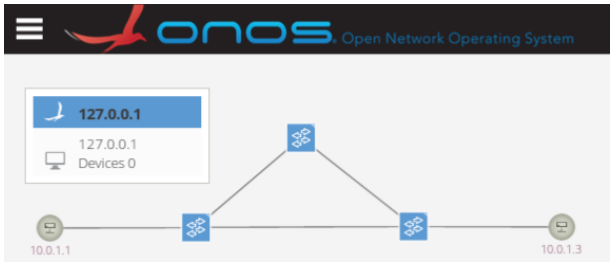
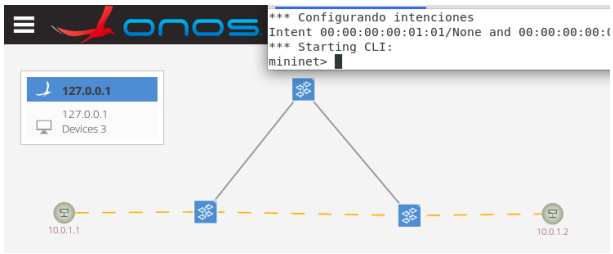
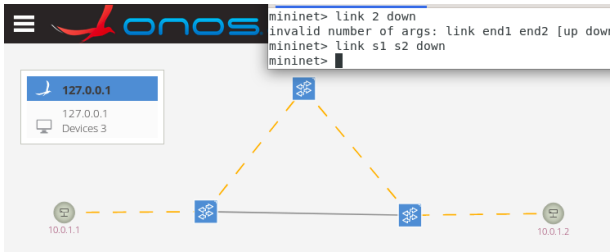


Figura 5.3: Subred óptica en la que integrar el servicio cuántico. En la imagen, su topología desplegada en Mininet y controlada por ONOS.

La interacción del controlador ONOS con Mininet es prácticamente autónoma en experimentos sencillos, pero se exploran despliegues más complejos para poder tener un control más ajustado del escenario. Así, se recurre a primitivas de nivel más bajo en Mininet y se desactivan las aplicaciones de ONOS que automatizan parte de la gestión. Por ejemplo, ONOS ejecuta de forma automática una aplicación que escucha el tráfico de resolución de direcciones que usa Ethernet, ARP, para descubrir las intenciones de los equipos e instalar las reglas de reenvío en consecuencia. Se decide, pues, sustituir este mecanismo por peticiones explícitas a la API REST de ONOS para informarle de esas intenciones, como se ejemplifica en la Figura 5.4. También se codifican otras llamadas a la API en el código de despliegue en Mininet para reiniciar el estado de ONOS o informar *a priori* de qué terminales existen en la red.



(a) Primera ruta.



(b) Segunda ruta.

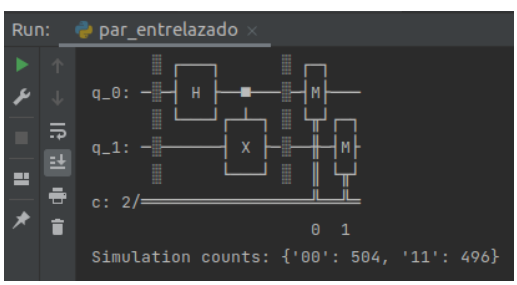
Figura 5.4: En esta captura se observa el comportamiento del controlador cuando se le informa de una intención. Al hacerlo, este resuelve la primera ruta para conectar los dos terminales. Sin embargo, si ese enlace se cae, el controlador se percata y resuelve una segunda ruta alternativa. Es interesante observar la nomenclatura que se usa en este controlador de redes definidas por *software*: las intenciones se resuelven y se «instalan» en la infraestructura.

Para las sucesivas fases, se modela la subred con el lenguaje NetAb y se crea con Acceleo un generador de código que sintetice de forma automática el despliegue para Mininet en Python. En la especificación recogida en el anexo «B Ejemplo de red programado con NetAb» aparece

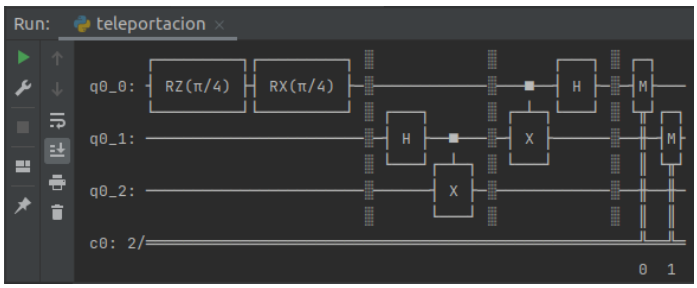
sintaxis del tipo @CLAVE=VALOR. Esta sintaxis se había añadido en un principio al lenguaje para facilitar la generación de código del modelo Transport API, que construye algunos de sus atributos con parejas de ese tipo; ha sido útil, sin embargo, para especificar la dirección IP de cada equipo.

5.2.2. Subred cúantica

De forma análoga, se codifica un programa en Python muy sencillo que ataca los procesadores cuánticos de la IBM para simular un canal de esa naturaleza basado en [27] y [28]. En concreto, se usa la biblioteca Qiskit para sintetizar un canal de transporte cuántico basado en el técnica de la teleportación, que permite transferir la información de un símbolo cuántico a otro con el fenómeno del entrelazamiento de estados. Para la simulación se usan las máquinas remotas de la IBM, por lo que se precisa conexión a Internet. En todo caso, estos procesadores tienen los registros limitados a unos pocos cúbits, por lo que no es posible hacer grandes ejecuciones. En la Figura 5.5 se muestran dos ejecuciones con esta biblioteca.



(a) Circuito para conseguir entrelazar un par de cúbits. La teoría garantiza que si en un extremo de ese par se observa un símbolo '0' o '1', en el otro extremo se mide el mismo, como se puede observar en las 1000 ejecuciones.



(b) Circuito que modela una teleportación. La información cuántica contenida en el primer registro, en este caso dos rotaciones de fase, se podrá medir en el tercer registro tras hacer colapsar los dos primeros.

Figura 5.5: Ejemplos de ejecuciones cuánticas en las máquinas de la IBM con la biblioteca Qiskit que se usarán en el enlace simulado.

Se idea una arquitectura compuesta de dos clientes y un servidor contra el que transaccionar; los primeros se ejecutarán dentro de la Mininet mientras que el segundo en la máquina anfitriona para poder comunicarse con la IBM. Así, mediante comandos simples de texto, y a través de comunicaciones contra los *sockets* de cada entidad, se coordina la ejecución remota en las máquinas cuánticas de los circuitos de teleporte y la obtención de los resultados.

5.2.3. Transport API

El soporte normativo de la ONF para este modelo es una especificación YANG [14], un lenguaje de modelado, pero además se ofrece una especificación OpenAPI, evolución de otra conocida como Swagger.[1] La primera describe el modelo de información propiamente dicho, su tipado y su política de acceso —*e. g.* solo lectura—; es la descripción normativa que debe, por ejemplo, una base de datos que lo almacene. La segunda describe una interfaz tipo REST

[1] Véase `https://editor.swagger.io/`

que respeta la primera, y se puede alimentar con ella generadores de código que crean tanto la estructura de clases que representa el modelo como sus métodos de acceso.

Mediante las definiciones OpenAPI se generan el esqueleto de un servidor al que las subredes informan de su topología y servicio con sendos clientes —también generados de la misma manera—. Estas piezas de código tienen una subrutina por cada una de las acciones que se puede hacer contra el modelo, por lo que resulta en un esqueleto de cientos de métodos. Además, se genera una clase por cada una de las entidades que representa el modelo, como se ilustra en la Figura 5.6, de tal forma que ante una llamada a la API REST se entrega un objeto en memoria que contiene la información de la petición —y viceversa, para generar una llamada se le ha de entregar al programa la instancia de uno de esos objetos—. De esa forma, se gestiona la recepción de peticiones y generación de respuestas, asegurándose que la estructura de información que se le provee y su tipado son acordes al modelo. Debido a su extensión, solo se programa la persistencia y tratamiento de un puñado de funciones, las más generales que permiten informar del modelo en su totalidad, y no de cada uno de sus detalles por separado. En el caso de la red óptica, la información se obtiene transaccionando contra la API REST de ONOS, mientras que en el caso de la red cuántica, es el propio código basado en Qiskit el que lo genera.

```
module model {
  yang-version 1.1;
  namespace "urn:model";
  prefix model;

  import ietf-yang-types { prefix "yang"; }
  description "A model";

  container context {
    description
      "A model context";

    container elements {
      uses model_elements;
      description "List of model elements";
    }
```

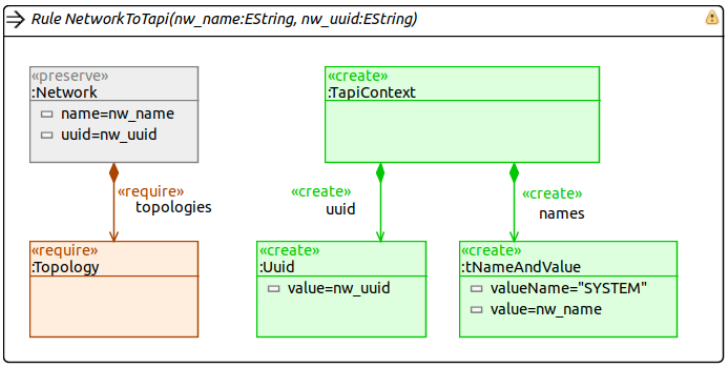
(a) Un modelo de ejemplo descrito con el lenguaje de modelado YANG, que especifica dos entidades, CONTEXT y ELEMENTS, una que cuelga de la otra.

```
# -*- coding: utf-8 -*-
import ...

# PY3 support of some PY2 keywords (needs improved)
if six.PY3:...
elif six.PY2:
  import __builtin__


class yc_elements_model__context_elements(PybindBase):...


class yc_context_model__context(PybindBase):...
```

(b) La implementación de ese modelo, con dos clases que representan cada entidad. Cada una tiene los métodos necesarios para manejar el acceso a sus atributos y para serializar y deserializar modelos.

Figura 5.6: En esta figura se recoge un modelo YANG y su implementación tal y como la genera OpenAPI para exponer cómo funcionan los clientes y el servidor descritos en esta sección.

Por otro lado, se articula una transformación de modelos con la que obtener una representación acorde al metamodelo de Transport API de la red descrita con NetAb. El objetivo es, de esta forma, poder comparar el modelo T-API recolectado de la red desplegada con el prescrito por el modelo «matriz» a modo de evaluación. Esto se hace mediante el lenguaje ATL —Atlas Transformation Language—, que usa sentencias OCL para inspeccionar el modelo origen y generar así los objetos del nuevo modelo.



(a) Henshin.

```
rule NetworkToTapiContext {
    from
        nw: netab!Network (
            not nw.hasTapiFacet()
        )
    to
        tc: tapi_common!TapiContext (
            uuid <- new_uuid,
            names <- Sequence { new_name }
        ),
        new_uuid: tapi_common!Uuid (
            value <- nw.uuid
        )
        new_name: tapi_common!tNameAndValue (
            valueName <- 'SYSTEM',
            value <- nw.name
        )
}
```

(b) ATL.

Figura 5.7: Las dos herramientas de transformación de modelos mencionadas, una provista de un DSL visual y la otra de uno textual.

En el capítulo de diseño se había propuesto el uso de Henshin para la transformación de modelos, una herramienta similar a ATL pero que usa un lenguaje gráfico menos expresivo y que ha resultado insuficiente; en la figura 5.7 se puede observar la diferencia entre ambas. Destacar, además, que el metamodelo UML de Transport API —también normativo, como las descripciones YANG— se ha tenido que transformar a Ecore de forma manual para poder usarse en Eclipse, puesto que de forma automática se perdía información.

5.3. Fase 2

Para esta segunda fase se programan los dos clientes que inician la transmisión cuántica desde los terminales simulados de la Mininet. Además, para poder ejecutarlos es necesario solicitar a la API con Python que ejecute los comandos necesarios para dotarles de acceso a Internet. De esa forma, estos clientes cursarán su información de señalización por la subred óptica mientras que solicitarán a IBM el envío de la información cuántica por la otra subred; es el escenario que se representó en la Figura fig:ibmq.

A continuación se modela el escenario de la segunda fase con NetAb y se alimenta al generador de código basado en Acceleo para que sintetice los nuevos artefactos de código. Tanto ese generador como la transformación del modelo a Transport API son los mismos que los usados en la primera fase, con la salvedad de que se añade la ejecución mencionada en el párrafo anterior para dar acceso a Internet; el comportamiento en el modelo de la fase no se ve modificado.

Por último, se hace el despliegue, que genera en ONOS el diseño de la Figura 5.8, y se ejecutan los clientes.

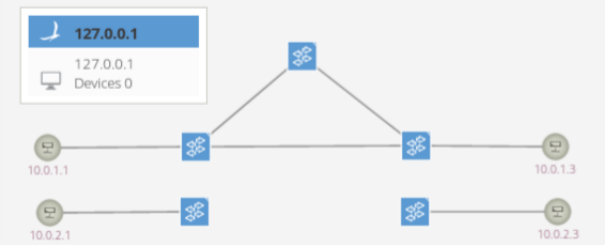


Figura 5.8: Captura de ONOS del despliegue en la segunda fase. En la subred cuántica, para sintetizar el enlace híbrido, los dos equipos de reenvío tienen salida al exterior para comunicarse con las máquinas de la IBM, aunque en esta imagen aparezcan sin una conexión entre ellos. Además, aunque se representan cuatro equipos terminales sólo deben considerarse dos, uno en cada extremo; ONOS representa cada interfaz como un equipo distinto.

5.4. Fase 3

Por última vez, se usan los generadores de código y la transformación de modelos para obtener los artefactos de *software* a partir de un modelo de la tercera fase descrito con NetAb.

Como se detalla en la Figura 5.9, esta última etapa culmina la integración de los dos servicios de transporte en una única red. En primer lugar, la infraestructura es única, con equipos de reenvío que encaminan tanto el tráfico óptico como el tráfico cuántico —de los equipos terminales al programa basado en Qiskit, que sigue simulando las transmisiones cuánticas en sí—, mediante dos subredes IP. La existencia de puntos de acceso IP, además, unifica de alguna manera el acceso a los dos servicios, que puede describirse como dirección más puerto. En último lugar, la gestión de la parte configurable de la red se delega a un único agente, el controlador ONOS.

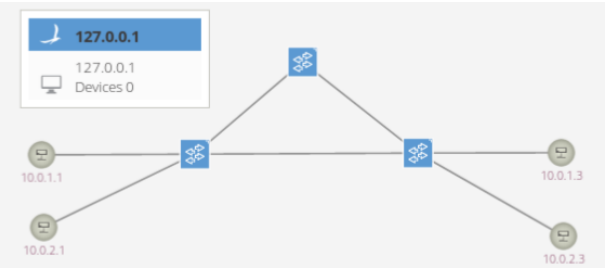


Figura 5.9: Tercera fase de la integración, con medios de reenvío compartidos. De nuevo, ONOS representa cada una de las interfaces de los equipos terminales como entidades separadas, pero sólo debe considerarse un equipo en cada extremo.

5.5. Conclusiones

Este capítulo ha expuesto los distintos desarrollos que se han hecho para articular el diseño propuesto de integración.

En primer lugar, se han detallado las tareas para preparar el entorno basado en ingeniería dirigida por modelos. Se ha modelado la sintaxis de un nuevo lenguaje, NetAb, y se ha usado para modelar las tres fases de la integración. Así, con herramientas del Eclipse Modelling Framework, se han hecho las transformaciones y síntesis de código adecuadas para desplegar y evaluar la integración.

En segundo lugar, se ha programado tanto el despliegue de la subred óptica con Mininet y ONOS como la simulación de las comunicaciones cuánticas con Qiskit y los agentes. Estos artefactos de código se han integrado también en el entorno de modelado para generarse en automático con la información del modelo de cada fase. Además, se ha generado los esqueletos tanto del servidor como de los clientes para la recolección del modelo de información Transport API usando su especificación OpenAPI, y se ha programado su funcionalidad.

Estos desarrollos permiten generar el flujo de trabajo propuesto para cada una de las tres fases de la integración. Es fundamental que entorno de modelado y los generadores de código son únicos para las tres fases; es un flujo dirigido por modelos, de tal forma que si se alimenta con la descripción de la fase enésima, se generan los artefactos de código adecuados para esa fase.

6

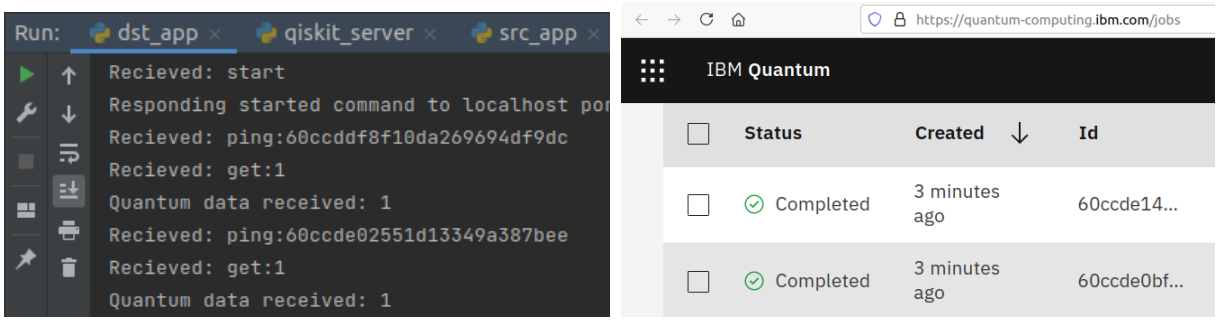
Resultados

En este capítulo se presentan las pruebas que se han hecho en los desarrollos para verificar que cumplen con lo prescrito en el diseño y la documentación previa.

6.1. Resultados parciales

En esta sección se quiere valorar algunos de los desarrollos necesarios para poder obtener el resultado global pero que no son de relevancia al evaluar aquel.

En primer lugar, en la Figura 6.1 se puede observar el sistema que se ha codificado para simular el enlace cuántico. Los dos clientes, uno de origen y otro de destino, transaccionan con un servidor central que gestiona la ejecución de los circuitos cuánticos en las máquinas remotas de la IBM. Esos circuitos son los que se habían ilustrado ya en la Figura 5.5 que ejecutan fenómenos de teleportación cuántica.



(a) Ejecución de los clientes cuánticos.

(b) Sitio web de IBM.

Figura 6.1: El enlace cuántico experimental se ha articulado con dos clientes y un servidor central que se comunica con las máquinas remotas de la IBM. Como consecuencia, los números de experimento que se observan en la ejecución de la izquierda son los mismos que detalla el sitio web de IBMQ en la derecha.

Y, en segundo lugar, se verifica que funcionan como deben los clientes y el servidor para la recolección del modelo Transport API. Estos se habían desarrollado a partir de los esqueletos de código que se sintetizan con las especificaciones OpenAPI publicadas junto con el modelo. Sin embargo, existía una gran parte del código que había que hacerla a mano, como son las rutinas de recolección de la información o las de guardado y recuperación en la base de datos. En la Figura 6.2 se muestran algunas líneas de código de la última rutina mencionada y, como

se puede observar, codificarlas se complica bastante en modelos tan extensos como Transport API. Esto se debe a que la estructura de objetos que se genera contiene muchas primitivas y de nombres muy largos. En todo caso, otra gran parte del código se genera en automático.

```
def data_tapi_commoncontext_tapi_topologytopology_context_nw_topology_service_topologytopology_uuid_get(topology_uuid):
    # Get the saved context and return the network topology service's topology reference if exists
    tapi_common_context = simple_db.load_from_db()
    tapi_topology_topology_context = tapi_common_context.topology_context
    topology_ref_wrapper = None
    if tapi_topology_topology_context:
        tapi_topology_topology_context_nw_topology_service = tapi_topology_topology_context.nw_topology_service
        if tapi_topology_topology_context_nw_topology_service:
            tapi_topology_topology_context_nw_topology_service_topologies = \
                tapi_topology_topology_context_nw_topology_service.topology
            if tapi_topology_topology_context_nw_topology_service_topologies:
                for each_topology_ref in tapi_topology_topology_context_nw_topology_service_topologies:
                    if str(each_topology_ref.topology_uuid) == str(topology_uuid):
                        topology_ref_wrapper = tapi.TapiTopologyTopologyRefWrapper(
                            topology=each_topology_ref
                        )
    print("[{}]: {}".format(
        TAPI_SERVER_SERVICE_DESCRIPTION,
        "Successful execution: " +
        data_tapi_commoncontext_tapi_topologytopology_context_nw_topology_service_topologytopology_uuid_get.__name__
    ))
    return topology_ref_wrapper
```

Figura 6.2: Algunas líneas de código que ha habido que programar para dotar de funcionalidad a los artefactos generados con la especificación OpenAPI para recolectar Transport API. En este caso se trata de la rutina para recuperar el modelo mediante un GET contra el servidor.

6.2. Proyecto de integración dirigida por modelos

En esta sección se evalúa el resultado final de la integración dirigida por modelos que se ha propuesto.

6.2.1. Modelado

El diseño comienza con el modelado de los tres escenarios propuestos con la sintaxis NetAb, puesto que es con ellos con los que se alimenta el resto del proceso. En la Figura 6.3 se muestra la captura de la instancia de Eclipse con la que se puede describir los tres escenarios, por lo que este ejercicio se asemeja al de *programar una red*. Al final, es mediante esta descripción mediante la cual se van a sintetizar los artefactos de código necesario para ello, aprovechando las técnicas de virtualización y de redes definidas por *software*.

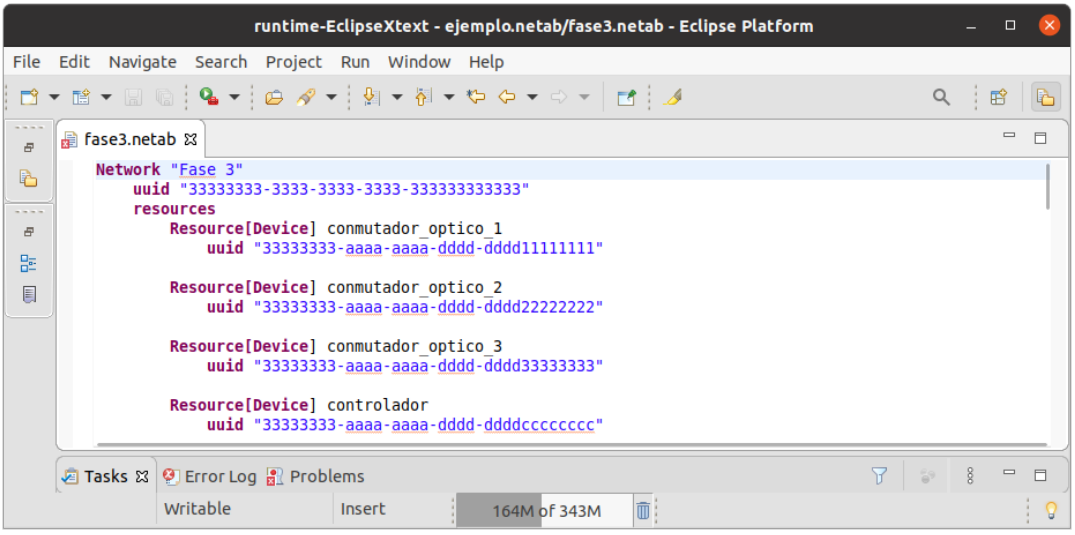


Figura 6.3: Captura de la instancia de Eclipse que, asistida por Xtext, permite programar una red con la sintaxis concreta textual de NetAb. Xtext genera los artefactos de código necesarios no sólo para generar la gramática con la que compilar estas descripciones a modelos XMI, sino, además, para señalar errores al programador o sugerir sus correcciones.

6.2.2. Transformación y generación de código

A continuación, se alimenta al resto de herramientas usadas en Eclipse Modelling Framweork para articular la cadena de producción de *software* diseñada. En primer lugar, se transforman los modelos que representan cada fase con el lenguaje ATL para obtener sus equivalentes en la sintaxis de Transport API. En segundo lugar, se usa Acceleo para generar el resto del código. Tanto la transformación como la síntesis de código es la esperada en las tres fases. En la Figura 6.4 se puede observar cómo cada ejecución de este tipo en Eclipse toma unas entradas, habitualmente modelos, y genera unas salidas, tanto otros modelos como código informático final.

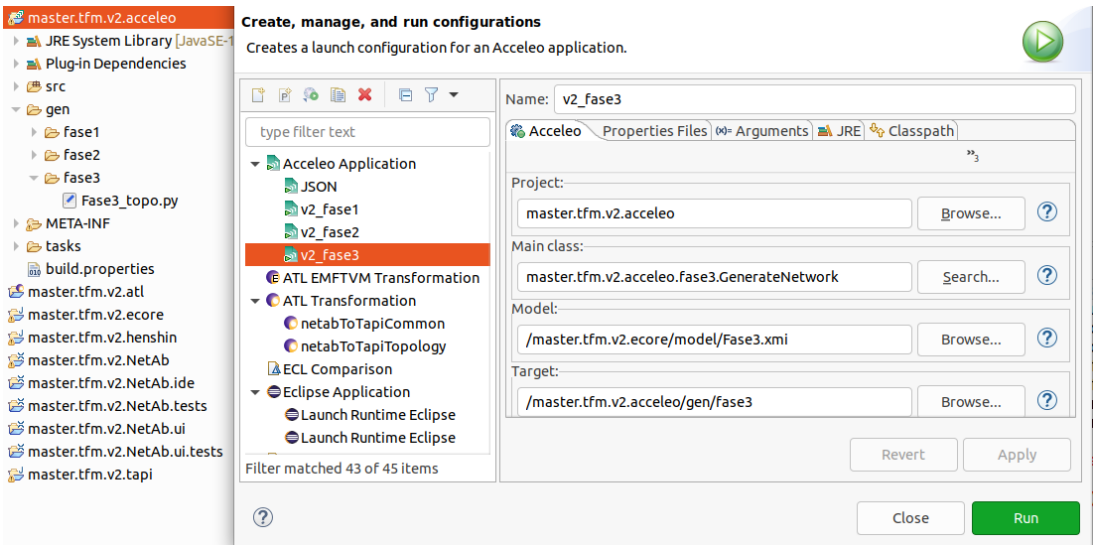


Figura 6.4: Interfaz de usuario para ejecutar las herramientas de Eclipse.

Para evaluar tanto lo descrito en esta sección como en la anterior se debe tener en cuenta que el diseño debería permitir modelar y programar cualquier red y no sólo las tres fases descritas. Las tecnologías dirigidas por modelos necesitan de unos desarrollos más aparatosos en sus etapas iniciales, puesto que es necesario crear el lenguaje de programación que se va a usar, las transformaciones y los generadores de código. Sin embargo, una vez hecho ese trabajo, la cadena compuesta puede ser alimentada por cualquier modelo siempre que se haya hecho con la debida diligencia. En este caso también se ha comprobado la veracidad de esas afirmaciones, puesto que ha sido posible completar las tres fases de integración con un único juego de transformaciones y generadores.

6.2.3. Funcionamiento

Una vez desplegada la red con los artefactos de código esta puede ser usada tanto para transmitir información tradicional por la red óptica como para las transmisiones cuánticas que generan los clientes desarrollados.

En el anterior capítulo ya se han recogido algunas capturas de la red en funcionamiento. Concretamente, se han presentado capturas de cómo el controlador ONOS representaba la topología de la red conforme se avanzaba de fase, así como de la instalación de intenciones.

En la Figura 6.5 se observa el escenario en funcionamiento durante la fase final de integración. Tal y como se habían diseñado los agentes que gobiernan el transporte cuántico, se usa la subred óptica para que transite su propia información de señalización, mientras que se usa la subred cuántica para comunicarse con Qiskit de IBM. Así, en esa imagen se puede observar tráfico de ambas naturalezas. La conectividad entre las interfaces de la red óptica se consigue mediante una intención que se informa al controlador en el despliegue. Sin embargo, la conectividad entre los equipos de reenvío y las interfaces de la subred cuántica se configura de forma programática, también en el despliegue de Mininet; los equipos de reenvío ejecutan sus procesos en el espacio de nombres de la máquina anfitriona, lo que hace posible la comunicación con Qiskit como se representó en la Figura 4.2

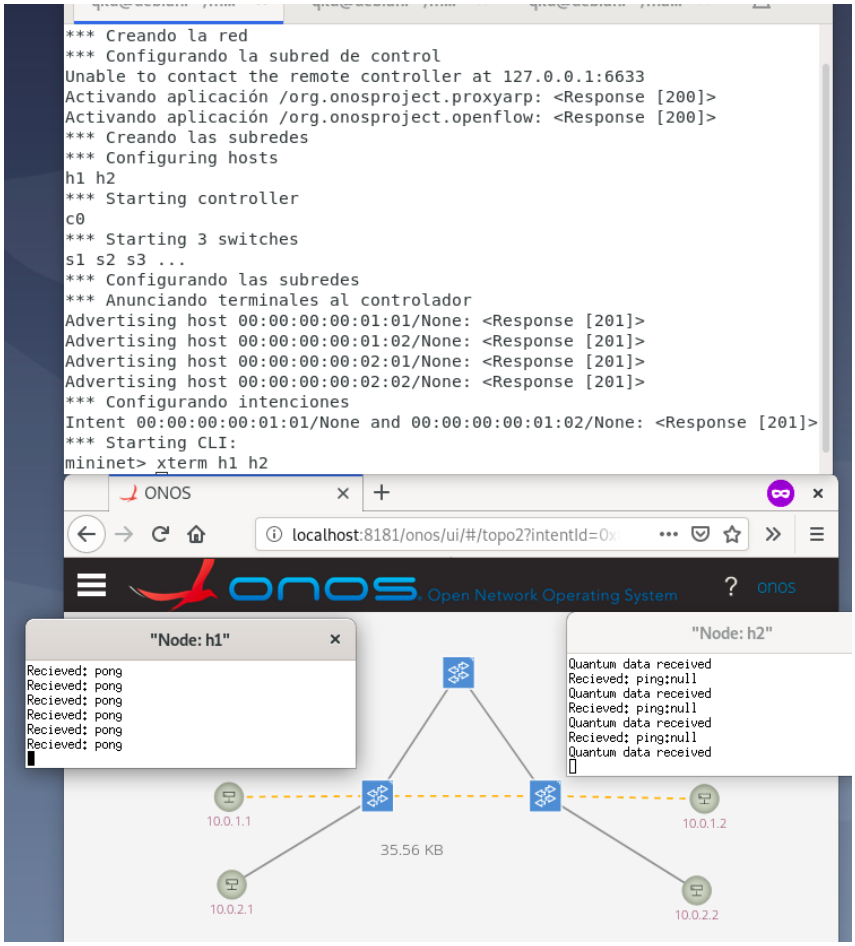


Figura 6.5: Última fase de integración en funcionamiento. En la consola superior se puede consultar la rutina de despliegue de Mininet, en la que se incluyen comunicaciones contra la API de ONOS. En la parte inferior, el propio controlador ONOS y dos consolas que se ejecutan en el espacio de nombres de los terminales de la red. En esas consolas se puede observa las salidas tanto del cliente origen como del cliente destino que coordinan el envío cuántico.

6.2.4. Evaluación I. Transport API

En esta sección se evalúa la hipótesis principal que se había formulado para sostener esta metodología de integración, que usando Transport API como modelo de abstracción era posible evaluar su evolución.

De esta manera, en la Figura 6.6 se observa el proceso de recolección del modelo de información. Después del despliegue, se levanta el servidor y se ejecuta de forma manual el cliente que recopila la información atacando la API REST del controlador ONOS. De esa forma, la información recolectada no surge de un código ideado para tal fin durante el despliegue, sino que es la información de gestión que conoce el controlador SDN. Por ello, la información que se tiene del enlace experimental cuántico, de naturaleza heterogénea, puede ser parcial, y se puede usar otro cliente para completarla.

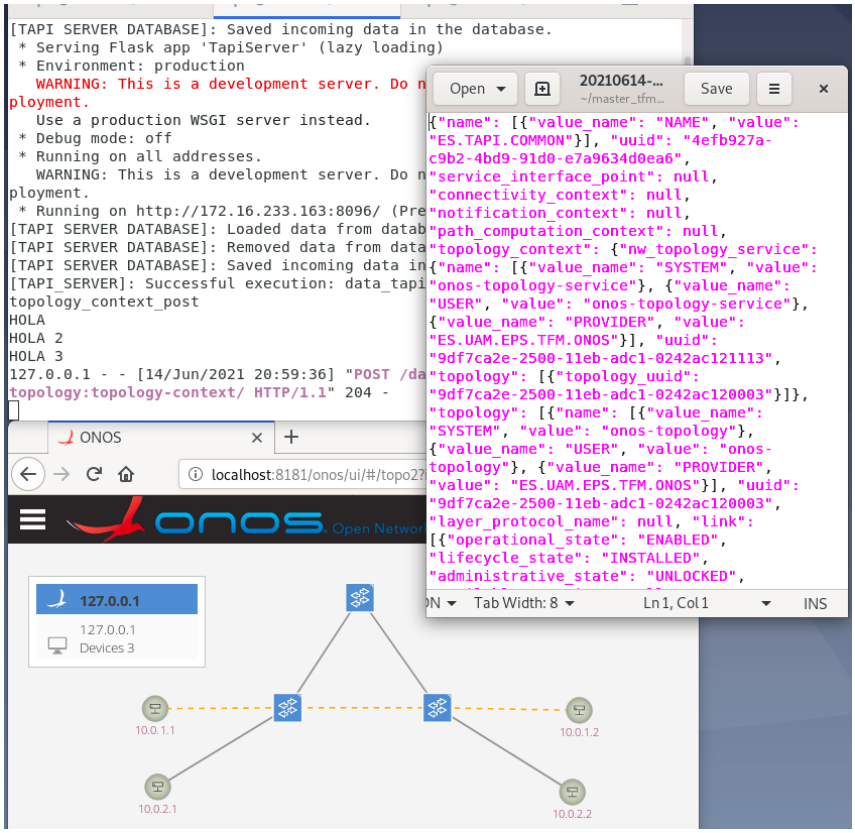


Figura 6.6: Proceso de recolección de la información que compone el contexto Transport API. En la consola de encima se ha ejecutado tanto el servidor como los clientes basados en OpenAPI que recopilan la información. En la captura permanece la información del servidor, que muestra una operación de tipo POST ejecutada de forma correcta. A su lado, se muestra el modelo recolectado, que se serializa en formato JSON y se persiste en memoria para su consulta.

Ese modelo se debe comparar con el generado con el propio modelo NetAb de la fase correspondiente, en este caso la tercera igualmente. De esa forma, se enfrenta el diseño prescriptivo con el descriptivo, y permite evaluar el grado de integración. En la Figura 6.7 se ilustra ese ejercicio, que se hace de forma manual. Se compara el modelo JSON recopilado con el modelo transformado en Eclipse desde la sintaxis de NetAb hasta las definiciones de Transport API; para ello se había descrito ya como se usa el lenguaje de transformación de modelos ATL.

Hay que mencionar que la inspección de los dos modelos presenta diferencias evidentes que, sin embargo, no refutan la hipótesis. Por ejemplo, el orden en el que aparecen los elementos de las colecciones es aribtrario, y se especifica como nulos ciertos módulos de Transport API que no se han implementado pero que la especificación de OpenAPI define de forma normativos. O, como se ilustra en la misma figura, se había escogido de forma consciente dar al modelo generado el nombre PROVIDER: NETAB mientras que al recopilado PROVIDER: ES.UAM.EPS.TFM.ONOS.

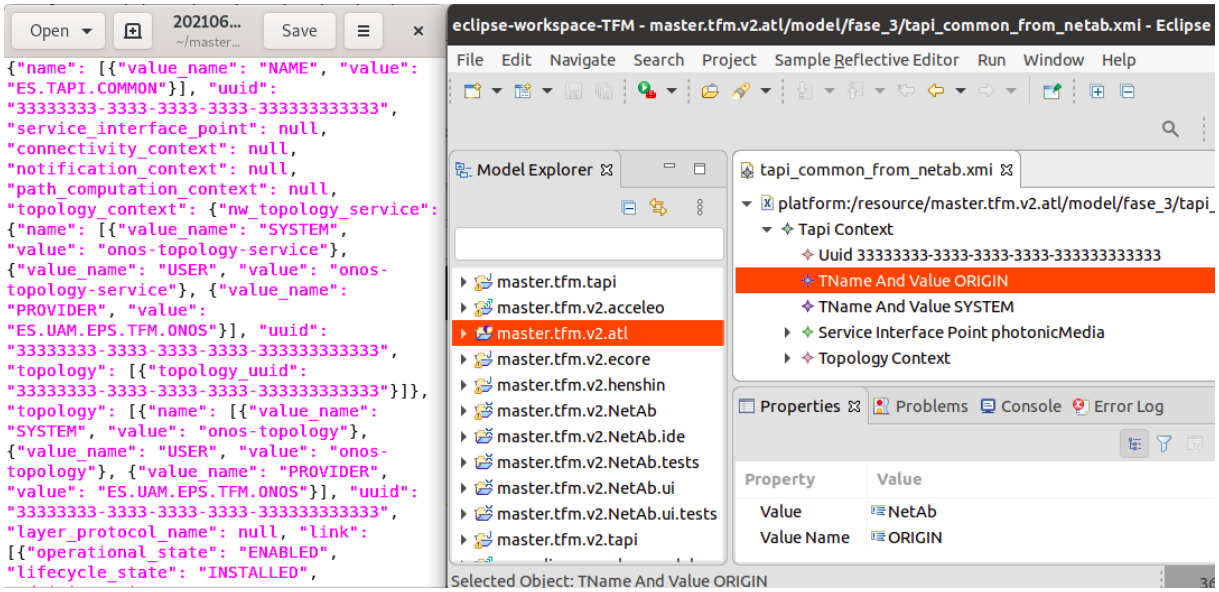


Figura 6.7: Comparación manual de los contextos Transport API esperado y recopilado. Se obervan ciertas diferencias como el nombre asignado al modelo o que algunos objetos son nulos en el modelo de información recolectado. Eso se debe a que OpenAPI genera sus artefactos de código usando una especificación que los define, por lo que espera su presencia.

6.2.5. Evaluación II. Pruebas

Por último, se ejecutan las pruebas que se generan de forma automática para evaluar tanto el grado de integración como si se cumple el diseño prescrito.

De esa forma, en la Figura 6.8 se puede observar la prueba basada en tráfico ICMP para comprobar la conectividad entre los dos terminales. En ese caso, y como era de esperar, sólo es posible alcanzar el otro terminal si se transita desde la interfaz de la subred óptica a su análoga. Esto permite garantizar que el tráfico de señalización entre los clientes está circulando efectivamente por la subred óptica.

Como se había indicado en el diseño, esta prueba es sólo una muestra de la capacidad del procedimiento dirigido por modelos de generar cuantos artefactos de código sean necesarios para desplegar, evaluar y operar la red. No es relevante, en este documento, el rendimiento final de la red desplegada, puesto que el objeto de discusión es el procedimiento dirigido por modelos.

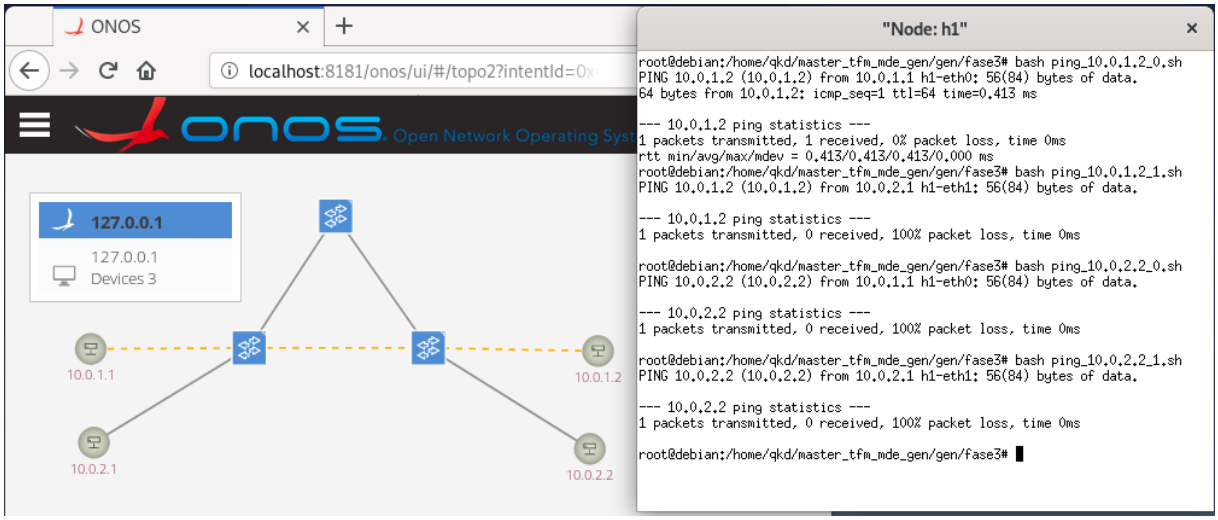


Figura 6.8: Pruebas de tráfico de paquetes ICMP para evaluar la conectividad. En este caso, se evalúa la conectividad entre los dos extremos, y se constata que desde un extremo sólo se alcanza el otro si se transita por la interfaz adecuada e intentando alcanzar el destino deseado. Es un ejemplo de artefacto de código que se puede generar para evaluar u operar la red.

6.3. Conclusiones

En este capítulo se constata que los desarrollos hechos para el proyecto cumplen con lo prescrito en el diseño y la documentación previa.

Como resultados previos, se ha revisado el código con el que se ha simulado el enlace cuántico, que usa las máquinas remotas de la IBM como se había diseñado, y los artefactos para recolectar el modelo Transport API basado en las especificaciones OpenAPI.

Como resultado global, se ha presentado el modelado de las fases de integración de la red; la generación de los artefactos de código necesarios para su despliegue y evaluación; así como la red en funcionamiento una vez integrada. El enlace cuántico experimental opera conjunto con la red óptica y la usa para enviar su señalización. Además, se verifica la conectividad deseada usando las pruebas que se habían generado para este caso usando tráfico ICMP. Se constata que este procedimiento permite articular la integración con tecnologías dirigidas por modelos. Y, su uso conjunto con las técnicas de virtualización y de SDN permiten hacer el ejercicio de *programar la red*.

Por último, se verifica la hipótesis que se había discutido en el diseño: mediante una capa de abstracción como Transport API es posible evaluar la evolución de la integración.

7

Conclusiones y trabajo futuro

En este capítulo se revisita la ejecución del proyecto para evaluar tanto su cumplimiento como lo acertado de las decisiones que se han tomado. Además, se detallan algunos desarrollos futuros que se han evaluado como posibles durante la misma.

7.1. Conclusiones

Del estado del arte Al documentar el estado del arte se ha hecho un esfuerzo adicional para poder extraer de la bibliografía un relato coherente qué refleje las líneas generales del diseño de redes de telecomunicación e informáticas durante las últimas décadas. Este ejercicio tenía como objetivo obtener una justificación o *rationale* de esa evolución, para poder tomar las decisiones de diseño más adecuadas en los desarrollos posteriores. De esa forma, se ha aprendido cómo esas redes no son sólo una infraestructura básica del contexto actual, sino que, además, han nacido y evolucionado con él. Se ha podido recopilar las principales decisiones de diseño, en especial aquellas para gestionar la complejidad mediante la abstracción, y relacionarlas con sus condicionantes tanto técnicos como políticos o económicos.

De forma adicional, esa evolución ha ratificado el papel que juega la informática para proveer a otras disciplinas de técnicas y tecnologías tanto para automatizar sus ingenios como para dotarlos de capas de abstracción. Tanto es así que se puede aventurar que ese potencial, junto con otras técnicas de la ingeniería del *software* como el modelado o la virtualización, permiten inaugurar en el diseño de sistemas heterogéneos una «ingeniería de la abstracción». Por ejemplo, este ejercicio se ha podido observar en la construcción de servicios sobre la red óptica mediante el apilado de varios juegos de niveles OSI —Figura 2.6—. Esa metodología delegaba la complejidad del servicio en los usuarios de la red y, con ella, el coste de sus propias decisiones de diseño. Los modelos de gestión dirigidos por modelos o el paradigma SDN permitirían también crear esos ingenios de abstracción.

Otra conclusión más concreta se extrae del estudio de la tecnología Ethernet, puesto que permite ejemplificar cómo la complejidad de un sistema no sólo reside en las decisiones implícitas en su diseño, sino también en aquellas explícitas en su configuración, administración u operación. Esta tecnología se había diseñado para descargar a la red de su complejidad en la capa de enlace, con medios de transmisión compartidos y técnicas difusivas. Sin embargo, al instalarlo con una configuración distinta a la prevista, esos mecanismos permiten ejecutar otras funciones con las que asumir la complejidad de otras tecnologías. Por ejemplo, con la configuración moderna en forma de malla, el encaminamiento IP puede ser sustituido con el ejecutado por Ethernet para resolver las direcciones MAC. El diseñador, por lo tanto, debe tener en cuenta ambas vertientes.

Por otro lado, la bibliografía en general, y las fuentes [11, 18] en particular, no describen el paradigma de redes definidas por *software* como aquellas en las que se favorece su programabilidad mediante técnicas informáticas, sino que centran su definición en dos detalles: la separación entre el plano de control y de datos y cierta centralización en las funciones de control. Sin embargo, se ha constatado como MPLS ya promovía esa segregación y permitía también centralizar las funciones de encaminado. Además, la decisión de centralizar cierto recurso o función en la red debe surgir de un ejercicio de proyección; en las redes ópticas jerárquicas SDH, por ejemplo, la descentralización era un acierto que permitía tiempos de recuperación ínfimos. Como consecuencia, parece desafortunado reducir el paradigma SDN a esos dos detalles, y no parece racional prescribir decisiones de diseño críticas. En cierta forma, quizás por un ejercicio académico, se pretende contraponer las redes SDN al paradigma clásico del Internet primigenio basado en TCP/IP, que era descentralizado y agregado. Esto permite explicar cómo se puede favorecer o facilitar ciertas características más o menos deseables en cada caso, como es en efecto la centralización, pero que son también la interoperatividad, la virtualización, la eficiencia, la modularidad… El potencial del paradigma SDN sólo se puede entender en su totalidad si se describe lo adecuado de dotar a las funciones de red de envolturas informáticas para favorecer su abstracción y programabilidad, sin imponer condiciones apriorísticas en el diseño.

Por último, se ha constatado cómo el uso de redes definidas por *software*, la ingeniería dirigida por modelos y las interfaces abiertas son tendencia en el diseño y gestión de redes. Además, se ha comprobado cómo las técnicas para abordar nuevos paradigmas, como el tránsito de información cuántica, pueden no ser muy distintas de las tradicionales, aunque se han de atender algunas peculiaridades como la benevolencia de los medios de transmisión con nuevos recursos o la gran variedad de tecnologías posibles: fotónica, microondas, crioelectrónica, espintrónica… Conforme este tipo de tecnologías se vayan desarrollando, la disciplina de la Telecomunicación va a necesitar tanto asimilar esas peculiaridades como proveer a los nuevos sistemas de los paradigmas, técnicas y tecnologías para el transporte efectivo y eficiente de la información.

Del diseño En la fase de diseño se debían haber tomado las decisiones adecuadas en la integración propuesta usando como base la justificación obtenida en los capítulos de documentación del estado del arte.

De nuevo, hay que destacar que se priorizaba el uso de tecnologías orientadas por modelos, pero se había documentado lo adecuado de esa elección. En primer lugar, se ha usado el ecosistema de modelado del proyecto Eclipse y, en segundo lugar, un lenguaje de dominio específico. Elegir el primero se evalúa como un acierto, debido a que permite usar de forma conjunta procesos muy potentes del desarrollo de código informático pero de con una dificultad asequible. Por el contrario, se pone en duda que para un proyecto pequeño compense crear la sintaxis textual de NetAb, puesto que Eclipse provee de métodos nativos para crear modelos como el representado en la Figura 7.1. En todo caso, tener un lenguaje con el que poder describir cualquier tipo de red es de mucha utilidad, y el valor didáctico de abstraer y axiomatizar todo un dominio de definición es extraordinario.

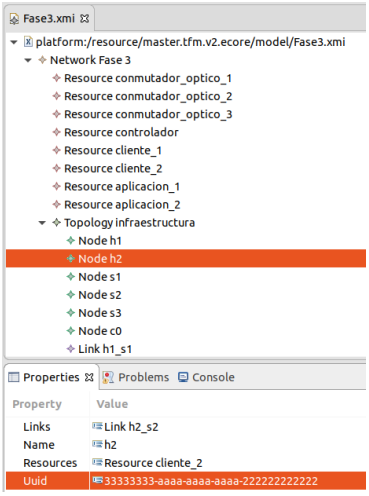


Figura 7.1: Herramienta de modelado basado en árboles que usa Eclipse por defecto.

Respecto al lenguaje, *a priori* se asume que con la sintaxis es posible expresar todos los escenarios que se desean, pero en la práctica se introducen nuevas

definiciones en las sucesivas manipulaciones del modelo, como en las transformaciones o en la generación de código. Por ello, hay que ser cuidadoso de cómo y dónde se introducen, puesto que de otra forma pueden ser erosiones de la arquitectura ideada en un comienzo. Un ejemplo claro de ello es la síntesis de código hecha con Acceleo, que se basa en el uso de plantillas, por lo que todas aquellas sentencias de código que no se generen a partir del modelo contienen definiciones semánticas que pueden no estar prescritas en el diseño. También se quiere poner de manifiesto lo complicado que puede llegar a ser usar el lenguaje ideado si no se conoce toda sus sintaxis concreta, puesto que se compone de muchísimas palabras clave, lo que dificulta su adopción si no es para grandes desarrollos. Esta característica no es, sin embargo, propia del DSL propuesto, y debe pensarse en la sintaxis que usa HTML o Latex, por poner ejemplos de un dominio específico como es la maquetación.

Respecto a las decisiones de diseño, en la recopilación que se había hecho en la documentación se ponía de manifiesto

- lo fundamental de las decisiones que se toman en el diseño de cada red y
- cómo con ellas se gestiona el reparto de la complejidad.

Y este ejercicio no se ha podido realizar en toda su extensión. La elección de tecnologías preexistentes es un condicionante y, por ejemplo, usar la pareja ONOS+Mininet determina transmisiones de paquetes, no orientadas a la conexión, conmutadas por reglas, de gestión centralizada… En todo caso, son herramientas muy adecuadas para desplegar de forma rápida y versátil redes programables y basadas en el paradigma TCP/IP.

Del desarrollo En el desarrollo, se consigue articular el flujo de trabajo propuesto, en la que cabe destacar que en las tres fases se usa la misma transformación de modelos y los mismos generadores de código —cada fase, alimentada con su propio modelo que la describe—. De esa forma, se constata que es posible generar una cadena de producción de código que permita desplegar redes usando técnicas de programación dirigida por modelos.

Han surgido algunos detalles como la dificultad de comparar de forma automática los contextos Transport API recopilado y esperado. Lo ideal hubiese sido contar con dos versiones cargadas en memoria para su inspección, pero mientras Eclipse trabaja en Java, los agentes desplegados se han codificado usando la síntesis que hace OpenAPI —basada a su vez en la biblioteca Pyangbind[1]— en Python; sólo un uso adecuado de estas herramientas permite anticipar estás necesidades. Por lo tanto, la comparación se ha hecho de forma manual.

De la evaluación Para evaluar la integración propiamente dicha se revisita la hipótesis hecha durante el diseño, «que al exponer las dos redes como una sola bajo una misma capa de abstracción, esta debe permanecer inmutable conforme el proyecto transite por sus fases». Y, en efecto, se ha constatado que eso es así y, fase tras fase, se ha generado un modelo de información Transport API equivalente. Además, la integración se ha evaluado como correcta ejecutando las pruebas propuestas.

Se desea indicar que este tipo de interfaces abiertas no sólo exponen un modelo de información, como se ha hecho, sino que además, como ocurría con la tecnología SNMP, la red debe poder configurarse mediante su modificación. Esta parte reactiva, en la que se ejecutan ciertas subrutinas de código cuando el modelo de información se actualiza, es mucho más compleja y no se ha realizado.

Además, cada modelo de información —o la interfaz abierta que modela— usa una sintaxis acotada para el propósito para el que fue modelada. Con esto se quiere poner en evidencia

[1] `https://github.com/robshakir/pyangbind`

que al escoger en el diseño qué modelo se usa para la abstracción, también se está escogiendo qué primitivas se van a evaluar —y con ello, qué semántica—. Por ejemplo, Transport API describe conceptos relativos a la topología, conectividad… en redes de transporte, por lo que al recolectar y comparar esa información sólo se puede garantizar que esos parámetros son inmutables conforme la integración avanza, pero no otros. Esta característica es inherente al hecho de que es una abstracción, por lo que se pierde cierta información que puede ser relevante. Es deseable, por ello, recolectar y comparar varios modelos.

Se ha de observar que, aunque el paradigma que gobierna el conjunto es fundamentalmente SDN, se ha de evaluar como un escenario heterogéneo en el que convive una red de transporte al uso con un enlace cuántico experimental. Es útil, para destacar esa diferencia, hacer una descripción basada en la abstracción y la complejidad como la que se ha hecho en capítulos anteriores. La red óptica se configura como cualquier sistema prototipo de comunicaciones informáticas definido por *software*, esto es, un segmento de reenvío con un control central que lo configura conforme al modelo TCP/IP. Sin embargo, y a diferencia de un entorno en producción, gran parte de la configuración se define con la ejecución de la Mininet, como son, en este caso, el plan de numeración IP. Por otro lado, la red cuántica aúna prácticamente toda su complejidad en el programa basado en Qiskit que simula la transmisión en los ordenadores de la IBM. Lo que sí que permite la capa de abstracción que se construye con el modelo de información Transport API es exponer estas dos redes como un único servicio de transporte homogéneo.

De esta forma, se puede afirmar que este método de trabajo permite gobernar los procesos de integración de dos redes de telecomunicación heterogéneas. Además, no es aventurado generalizar estos resultados y afirmar que es posible simular, desplegar, evaluar y operar redes de telecomunicación con esta metodología, generando los artefactos de código necesarios para ello. Por ejemplo, *a posteriori* se ha podido comprobar que en [29] se ha llegado a conclusiones similares.

Además, como se usan herramientas informáticas adecuadas para generar capas de abstracción, es posible usarlas para gestionar la complejidad en los diseños de redes con decisiones como las descritas en la documentación del estado del arte. Estas técnicas son la ingeniería dirigida por modelos, la abstracción programática, la virtualización y las interfaces abiertas.

7.2. Trabajo futuro

Una línea de investigación futura es indagar cómo sería la integración con una alternativa de diseño que se identificó durante los desarrollos. ONOS permite abordar esta integración mediante medios propios, con un diseño basado en el protocolo NETCONF. En concreto, debido a que en un caso de uso de la ONF se usó el Transport API como modelo de información,[2] su definición YANG se puede invocar en esta plataforma como una aplicación. Se debería poder alimentar al protocolo NETCONF con esta definición, de tal forma que ONOS fuese capaz de recopilar la información para modelo común directamente de los equipos con ese protocolo. Por último, sería necesario programar otra aplicación sobre ONOS capaz de procesar la información obtenida y generar las respuestas adecuadas.

También, las piezas de código que usan la librería Qiskit para controlar los ordenadores cuánticos de la IBM se basan en [27] y [28] para simular canales cuánticos, pero que se desea incluir lo descrito en [30] para desplegar una red con conmutación que permita distribuir entrelazamiento cuántico entre varios nodos.

Puesto que con la ingeniería dirigida por modelos se concatenan muchas tareas, se quiere

[2] `https://wiki.onosproject.org/pages/viewpage.action?pageId=23335851`

investigar cómo coordinarlos mediante algún robot de integración y despliegue continuos como Jenkins.[3]. De forma adicional, el entorno de modelado de Eclipse es muy adecuado para desarrollos académicos, pero en producción se usan otras tecnologías, como el lenguaje de modelado YANG o código escrito en Python; se desea explorar la viabilidad de trasladar el flujo de trabajo propuesto a un entorno en producción real.

Por último, se desea evaluar la usabilidad del lenguaje NetAb con nuevos usuarios, tanto expertos del dominio —telemáticos— como perfiles más trasversales —programadores, docentes…—. Este tipo de pruebas son habituales, por lo que existen múltiples métodos y métricas para hacerlo, como las descritas en [31].

[3] `https://www.jenkins.io/`

Anexos

Anexo A

Glosario

En este anexo se detallan algunos conceptos útiles para abordar la lectura del documento.

Abstracción La abstracción es una herramienta útil y transversal en ingeniería, sobre todo para gestionar la complejidad [32]. Mediante este proceso, se traslada una realidad a una descripción, tanto con el nivel de detalle como con el enfoque deseados. Es decir, permite no sólo modular el nivel de complejidad en una descripción, sino también detallar únicamente las descripciones útiles para cierto perfil. Como las soluciones que propone la ingeniería siempre son en cierto grado complejas, debido tanto a su dificultad como a lo críptico de sus descripciones, es fundamental tener las herramientas adecuadas para representarlas de clara pero inequívoca.

Además, los procesos de abstracción son naturales para la mente humana, capaz de reducir, generalizar, concretar, traducir… de forma natural. Las máquinas, por el contrario, necesitan de las técnicas y tecnologías para ello, ya sean implícitas en su diseño o mediante programación informática.

Heterogeneidad Un sistema es «heterogéneo» cuando funciona con paradigmas, técnicas, tecnologías o procesos distintos e, incluso, incompatibles sin la armonización adecuada. Por ejemplo, hoy en día, los sistemas electrónicos de lógica programable —conocidos por las siglas en inglés de la tecnología *field-programmable gate array*, FPGA— incorporan, de forma adicional a ese sustrato, microcontroladores, proceso de señal de radiofrecuencia, o buses de alta capacidad para hacer tareas gráficas o de inteligencia artificial; usan tecnología digital y analógica, con secciones en corriente continua y otras en alterna, etc.

Interoperatividad y normalización Muchas de las soluciones de la ingeniería deben poderse interoperar, es decir, deben incorporar una serie de características comunes que faciliten su uso conjunto. Por ello, en escenarios heterogéneos, la normalización tiene un papel fundamental para discutir, armonizar y publicar definiciones comunes con las que diseñar estos sistemas. Los procesos normativos tienen otros beneficios, como lograr una economía de escala que abarate los procesos productivos como consecuencia de la uniformidad de los diseños.

En todo caso, es especialmente relevante en este documento la normalización de interfaces, que se suelen llamar entonces «interfaces abiertas» —publicadas, del inglés *open*—; la de procesos, o la de evaluaciones, tiene menos interés en este caso. Para que dos sistemas sean compatibles debe mediar entre ellos una conexión común en sus diseños, mientras que el resto puede ser muy heterogéneo. En ocasiones este tipo de interfaces no son estándares industriales, sino propietarios de una organización que decide publicarla. Sea como sea, se trata de un ejercicio de abstracción para encontrar un espacio común.

Nivel	Función	Ejemplos
Físico	Permitir las transmisiones	Medio, adaptación, interfaces…
Enlace de datos	Garantizar las transmisiones	Concurrencia, corrección de errores…
Red	Garantizar la tránsito y entrega	Plan de numeración, congestión…
Transporte	Garantizar la integridad	Segmentación, reconstrucción, reenvío…
Sesión	Gestionar la vida de la conexión	Identificación, bidireccionalidad, pausado…
Presentación	Adaptar los mensajes	Codificación, *big endian vs. little endian*…
Aplicación	Dotar funcionalidades a los usuario	Protocolos, programas…

Tabla A.1: Niveles del modelo OSI.

En las telecomunicaciones, como en otras disciplinas cercanas, es de especial relevancia el modelo de interconexión de sistemas abiertos, una norma conjunta de la UIT y la ISO más conocida por sus siglas en inglés OSI, de Open Systems Interconnection. Esta norma divide las funciones de la red en siete niveles para poder acotar en qué tareas trabaja cada uno de sus equipos, como de describe en la Tabla A.1. Su objetivo es lograr la interoperatividad, por lo que si un equipo se certifica para ejecutar funciones del nivel enésimo en una determinada red, debe poder operar tanto con equipos de ese mismo estrato como de los adyacentes.

Arquitectura En la ingeniería del *software*, la «arquitectura» que posee un código informático es el conjunto de decisiones de diseño previas al desarrollo del programa, como son, por ejemplo, las relativas a su estructura, su comportamiento funcional o sus dependencias y conexiones. El diseño de la arquitectura de un programa es un proceso previo a su codificado en el que se proyecta su estructura y funcionalidad mediante procesos de abstracción; se puede encontrar una explicación mucho más precisa en [33].

Se denomina «arquitectura prescriptiva» a las características del código que se proyectan de forma previa a su escritura, mientras que se llama «arquitectura descriptiva» a las características finales. Por último, se denomina «erosión» al proceso por el cuál la descripción final no recoge todas las decisiones de diseño impuestas por la prescripción inicial.

Interfaces programáticas Cuando hay que integrar un proyecto informático con otro, o bien se publica el código o se diseña una «arquitectura adaptativa» [33]. Esta segunda opción es la elegida cuando se busca que el programa sea percibido como un producto monolítico del que se desconocen sus mecanismos. Son diseños de arquitecturas adaptativas, por ejemplo, el de la «interfaz de programación de aplicaciones», o API por sus siglas en inglés. Este tipo de conectores expone un punto de acceso a ciertas subrutinas del código para interactuar con su funcionalidad. Otras arquitecturas explotan la «comunicación entre procesos», IPC en inglés, en las que se delega toda la ejecución al código encapsulado —como las «llamadas a procedimientos remotos», RPC—.

Además, mediante este tipo de arquitecturas se pueden tomar decisiones de diseño sobre qué nivel de abstracción se desea otorgar a un código informático. Por ejemplo, en primer lugar las funciones que se exponen en una API determinan que puntos de acceso al programa se tienen. Y, en segundo lugar, dos códigos escritos de forma muy distinta pueden exponerse con dos API de funcionamiento idéntico.

Como consecuencia, si se logra dotar a una solución de cualquier otra disciplina con una envoltura informática que use estas técnicas, su integración e interoperatividad con otras puede llegar a ser trivial.

Virtualización La virtualización es una técnica informática que permite repartir los recursos de los que dispone un sistema informático en distintos entornos, de tal forma que los programas que se ejecutan en cada uno de ellos los perciben como si se tratase de una única máquina. Como consecuencia, en una misma máquina física pueden ejecutarse múltiples equipos virtuales como encaminadores, *firewalls*, máquinas servidoras… En el caso de tratarse, como en estos ejemplos, de máquinas que ejecutan funciones de red la técnica se denomina virtualización de funciones de red, NFV por sus siglas en inglés. El uso de esta técnica en redes tiene múltiples ventajas, como crear y destruir a equipos a discreción, sustituirlos por nuevas versiones o cambiar de proveedor sin tener que actualizar todo el inventario. Como desventaja, se asumen los riesgos propios de usar sistemas más informatizados, como las posibles vulnerabilidades de seguridad o inestabilidades.

Lenguajes A lo largo de este documento se mencionan algunos conceptos relacionados con la lingüística, puesto que la informática es una disciplina muy cercana a estas definiciones y conviene detallar algunas. Un lenguaje se compone de una serie de símbolos para poder ser expresivo, pero estos no pueden ser usados de forma arbitraria, sino que poseen una estructura lógica que los relaciona, su «sintaxis». Así, es posible usar los símbolos *la* y *casa*, pero sólo en la secuencia *la casa* y no en *casa la*. Además, un símbolo puede ser escrito, pero también se puede hablar o gestualizar, por lo que un lenguaje no se compone únicamente con ellos, si no con una serie de ideas que encierran detrás. El concepto o idea de "*casa*" forma parte de la sintaxis «abstracta» del lenguaje, pero su grafía, su palabra hablada, su dibujo o su gesto en lenguaje de signos forman parte de distintas sintaxis «concretas». La «semántica» de ese lenguaje es, por último, el significado que evoca la sintaxis y, así, la secuencia *la casa* puede ser un inmueble o un linaje monárquico.

Modelos Una de las técnicas más adecuadas para construir abstracciones son los «modelos», representaciones esquemáticas que describen un sistema o una de sus abstracciones. Son de interés debido a que son herramientas muy expresivas y, por ello, ideales para usarlos no sólo en el desarrollo de programas informáticos, sino como programas informáticos en sí. Pero además, precisamente por esta capacidad de expresión, es posible usar un modelo para describir por sí sólo una sintaxis, por lo que se puede definir con ellos un lenguaje; en ese caso se denominan «meta-modelos».

Son muy útiles para representar semánticas más reducidas que las que posee un lenguaje de uso general, de tal forma que con un modelo se puede definir la sintaxis de un «lenguaje de dominio específico», esto es, un conjunto reducido de conceptos con los que poder escribir programas en un campo semántico concreto.

Técnicas de telecomunicación La Teoría de la comunicación detalla un conjunto de técnicas con las que construir sistemas de telecomunicación sobre cualquier medio físico. Cabe mencionar un conjunto de ellas que se mencionan en el documento, sin entrar en más detalle.

- La «codificación» permite asociar a cada evento del que se puede extraer cierta información —como leer una letra o detectar la presencia de una bacteria— de un símbolo lógico para representarlo de forma inequívoca, incluso en presencia de distorsiones y ruidos.
- La «modulación» hace corresponder cada uno de los símbolos lógicos en una transmisión con un conjunto de señales físicas adecuadas tanto para el medio usado como para proteger a la información frente al ruido.
- La «multiplexación» permite el envío de varias transmisiones en un único medio físico segmentando sus recursos.

Funciones	**Detalle**
del plano de datos	Ejecución de la lógica de red
del plano de control	Lógica de red para conmutar, encaminar…
del plano de gestión	Configuración, mantenimiento, recogida de información…
del plano de aplicación	Funciones de alto nivel como análisis, reglas de negocio…

Tabla A.2: Estratos en los que se puede dividir las funciones de red.

- El «acceso múltiple» coordina la concurrencia de varios transmisores en un único medio físico.
- La «duplexación» garantiza la concurrencia de los transmisores necesarios para construir telecomunicaciones bidireccionales.
- La «conmutación» selecciona el medio físico por el que debe transcurrir la transmisión en una red para alcanzar su destino.

Circuitos y paquetes En una transmisión se han de gestionar una serie de recursos físicos, por lo que tiene especial relevancia la técnica usada para aprovisionarlos.

- En el aprovisionamiento de «circuitos» la red reserva para cada transmisión todos los recursos necesarios de un extremo a otro en el momento de establecerse y los usa en exclusiva hasta que termina.
- En el aprovisionamiento de «paquetes», por el contrario, la información fluye por la red encapsulada de alguna forma y es posible que solo use los recursos por los que transita en cada momento, que se asignan y liberan de forma dinámica.

Por último, se llama «circuito virtual» a un aprovisionamiento en el cuál cada transmisión tiene reservado una serie de recursos en un entramado que se asemeja mucho al concepto de paquete. Sin embargo, a diferencia de este, esas tramas se cursan de forma regular, coordinada y garantizada.

Es habitual detallar que el funcionamiento de circuitos es «orientado a la conexión», es decir, que se ofrece cierta garantía a los usuarios de entrega de la información. Por el contrario, al detallar el paradigma de paquetes suele usarse el término «*best effort*», que se refiere a aquellas técnicas en las que la disponibilidad de los recursos no se garantiza para cada transmisión aislada, y por ello los usuarios de la red han de asumir que los recursos son limitados y puede existir disfunciones en la red. No son, sin embargo, definiciones ni excluyentes ni contrarias.

Redes de telecomunicación Las redes de telecomunicación son estructuras en las que se enlazan mediante medios de transmisión una serie de nodos interconectados en forma de grafo. En ella, se ejecutan lo que se denomina «funciones de red», esto es, cada uno de los mecanismos necesarios para el tránsito de información. En el diseño, es fundamental cómo se distribuyan entre los subsistemas, puesto que determina dónde reside más o menos complejidad y qué abstracciones se hacen. Son funciones de red, por ejemplo, la gestión de acceso al medio físico, la tarifación del servicio o el conmutado; en la tabla A.2 se puede encontrar una clasificación muy usada.

En este documento se hace uso también de la diferencia entre red de acceso, red troncal y red de transporte. La primera, se refiere a los extremos de la red, en la que están los terminales, mientras que la segunda hace referencia al núcleo que les da servicio. Por último, una red de transporte es aquella cuyas funciones de red son las propias para el tránsito efectivo de información, habitualmente de una red a otra.

Anexo B

Ejemplo de red programado con NetAb

```
Network Fase 3
  uuid 33333333-3333-3333-3333-333333333333
  resources
    Resource[Device] conmutador_optico_1
      uuid 33333333-aaaa-aaaa-dddd-dddd11111111
    Resource[Device] conmutador_optico_2
      uuid 33333333-aaaa-aaaa-dddd-dddd22222222
    Resource[Device] conmutador_optico_3
      uuid 33333333-aaaa-aaaa-dddd-dddd33333333
    Resource[Device] controlador
      uuid 33333333-aaaa-aaaa-dddd-ddddcccccccc
    Resource[Device] cliente_1
      uuid 33333333-aaaa-aaaa-dddd-ddddcccc1111
    Resource[Device] cliente_2
      uuid 33333333-aaaa-aaaa-dddd-ddddcccc2222
    Resource[Application] aplicacion_1
      uuid 33333333-bbbb-bbbb-dddd-ddddaaaa1111
    Resource[Application] aplicacion_2
      uuid 33333333-bbbb-bbbb-dddd-ddddaaaa2222

  slices
    Slice[Service] servicio_integrado
      uuid 33333333-cccc-cccc-cccc-cccccccccccc
      topologies
        servicio

  topologies
    Topology infraestructura
      uuid 33333333-aaaa-aaaa-aaaa-aaaaaaaaaaaa
      nodes
        @ip="10.0.1.1"
        @ip="10.0.2.1"
        Node h1
          uuid 33333333-aaaa-aaaa-aaaa-111111111111
          links h1_s1
          resources cliente_1
        @ip="10.0.1.3"
        @ip="10.0.2.3"
        Node h2
          uuid 33333333-aaaa-aaaa-aaaa-222222222222
          links h2_s2
          resources cliente_2
```

```
      @ip="10.0.1.2"
      @ip="10.0.2.2"
      @ip="10.0.1.5"
      @ip="10.0.1.6"
      @c_eth="ens33"
      Node s1
        uuid 33333333-aaaa-aaaa-aaaa-aaaa11111111
        links h1_s1, s1_s2
        resources conmutador_optico_1
      @ip="10.0.1.4"
      @ip="10.0.2.4"
      @ip="10.0.1.7"
      @ip="10.0.1.8"
      @c_eth="ens38"
      Node s2
        uuid 33333333-aaaa-aaaa-aaaa-aaaa22222222
        links h2_s2, s1_s2
        resources conmutador_optico_2
      @ip="10.0.1.9"
      @ip="10.0.1.10"
      Node s3
        uuid 33333333-aaaa-aaaa-aaaa-aaaa33333333
        links s2_s3, s3_s1
        resources conmutador_optico_3
      Node c0
        uuid 33333333-aaaa-aaaa-aaaa-aaaacccccccc
        resources controlador
    links
      Link h1_s1
        uuid 33333333-aaaa-aaaa-aaaa-111111aa1111
        nodes h1, s1
      Link h2_s2
        uuid 33333333-aaaa-aaaa-aaaa-222222aa2222
        nodes h2, s2
      Link s1_s2
        uuid 33333333-aaaa-aaaa-aaaa-aa1111aa2222
        nodes s1, s2
      Link s2_s3
        uuid 33333333-aaaa-aaaa-aaaa-aa2222aa3333
        nodes s2, s3
      Link s3_s1
        uuid 33333333-aaaa-aaaa-aaaa-aa3333aa1111
        nodes s3, s1
  Topology servicio
    uuid 33333333-bbbb-bbbb-bbbb-bbbbbbbbbbbb
    nodes
      Node s_h1
        uuid 33333333-bbbb-bbbb-bbbb-111111111111
        links s_h1_h2
        resources aplicacion_1
      Node s_h2
        uuid 33333333-bbbb-bbbb-bbbb-222222222222
        links s_h1_h2
        resources aplicacion_2
    links
      Link s_h1_h2
        uuid 33333333-bbbb-bbbb-bbbb-111111222222
        nodes s_h1, s_h2
```

```
Architecture sdn
  uuid 33333333-dddd-dddd-dddd-333333333333
  arrays
    Segment[Wired] acceso
      uuid 33333333-dddd-dddd-dddd-aaaaaaaaaaaa
      resources cliente_1, cliente_2, aplicacion_1, aplicacion_2

    Segment[Optical] transporte
      uuid 33333333-dddd-dddd-dddd-bbbbbbbbbbbb
      resources conmutador_optico_1, conmutador_optico_2,
            conmutador_optico_3, controlador

    Plane[Control] control
      uuid 33333333-dddd-dddd-dddd-cccccccccccc
      resources controlador

    Plane[Forwarding] datos
      uuid 33333333-dddd-dddd-dddd-dddddddddddd
      resources conmutador_optico_1, conmutador_optico_2,
            conmutador_optico_3, cliente_1, cliente_2

    Layer[Application] aplicacion
      uuid 33333333-dddd-dddd-dddd-eeeeeeeeeeee
      resources aplicacion_1, aplicacion_2
      topologies servicio
  abstractions
    Abstraction mininet
      uuid 33333333-dddd-dddd-aaaa-333333333333
      arrays
        control, datos

      Abstraction tapi_topologia
      uuid 33333333-dddd-dddd-bbbb-333333333333
      arrays
        control, datos

    Abstraction tapi_servicio
      uuid 33333333-dddd-dddd-cccc-333333333333
      arrays
        aplicacion
```

Bibliografía

[1] Adrian Farrel and Igor Bryskin. *GMPLS. Architecture and Applications*. Morgan Kaufmann, 2006. ISBN 9780120884223.

[2] Rajiv Ramaswami and Kumar N. Sivarajan. *Optical Networks. A practical perspective*. Morgan Kaufmann, 2002. ISBN 9781558606555.

[3] Fernando Sáez Vacas. Teleinformática, telemática, telebernética. *BIT*, May 1980. Colegio Oficial de Ingenieros de Telecomunicación.

[4] UIT-T. Recomendación x.25: Interfaz entre el equipo terminal de datos yel equipo de terminación del circuito de datospara equipos terminales que funcionan en elmodo paquete y están conectados a redespúblicas de datos por circuitos especializados. *Serie X*, 1996. Unioń Internacional de Telecomunicaciones.

[5] Vinod Joseph and Brett Chapman. *Deploying QoS for IP Next Generation Networks*. Morgan Kaufmann, 2009. ISBN 9780123744616.

[6] Charles Kalmanek. A retrospective view of atm. *SIGCOMM Comput. Commun. Rev.*, 32(5):13–19, November 2002. Association for Computing Machinery.

[7] Barry M. Leiner, Vinton G. Cerf, David D. Clark, Robert E. Kahn, Leonard Kleinrock, Daniel C. Lynch, Jon Postel, Larry G. Roberts, and Stephen Wolff. A brief history of the internet. *SIGCOMM Comput. Commun. Rev.*, 39(5):22–31, October 2009.

[8] Leonard Kleinrock. An early history of the internet [history of communications]. *IEEE Communications Magazine*, 48(8):26–36, 2010.

[9] Douglas E. Comer. *Automated Network Management Systems*. Pearson Prentice Hall, 2007. ISBN 9780132393085.

[10] Adrian Farrel, editor. *Network Management Know It All*. Morgan Kaufmann, 2009. ISBN 9780123745989.

[11] Open Networking Foundation. Sdn architecture overview, version 1.1, onf tr-504. 2014.

[12] Douglas C. Schmidt. Guest editor's introduction: Model-driven engineering. *Computer*, 39(2):25–31, February 2006. IEEE Computer Society Press.

[13] J. Medved, R. Varga, A. Tkacik, and K. Gray. Opendaylight: Towards a model-driven sdn controller architecture. In *Proceeding of IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks 2014*, pages 1–6, 2014.

[14] M. Bjorklund. YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF). RFC 6020, RFC Editor, Octubre 2010.

[15] R. Enns y M. Bjorklund. Network Configuration Protocol (NETCONF). RFC 6241, RFC Editor, Octubre 2010.

[16] Ramon Casellas, Ricard Vilalta, Ricardo Martinez, and Raúl Muñoz. *SDN Control of Disaggregated Optical Networks with OpenConfig and OpenROADM*, pages 452–464. 02 2020.

[17] Victor Lopez, Ricard Vilalta, Victor Uceda, Arturo Mayoral, Ramon Casellas, Ricardo Martinez, Raul Munoz, and Juan Pedro Fernandez Palacios. Transport api: A solution for sdn in carriers networks. In *ECOC 2016; 42nd European Conference on Optical Communication*, pages 1–3, 2016.

[18] A. S. Thyagaturu, A. Mercian, M. P. McGarry, M. Reisslein, and W. Kellerer. Software defined optical networks (sdons): A comprehensive survey. *IEEE Communications Surveys Tutorials*, 18(4):2738–2786, 2016.

[19] Makoto Murakami, Han Li, and Jeong dong Ryoo. Packet transport networks: overview and future direction. In *2ª APT/ITU Conformance and Interoperability Workshop 2014*, 2014. Accesible en `https://www.itu.int/en/ITU-T/C-I/interop/Documents/20140826/CI-2_INP-09_Packet_Transport_Networks_Overview_and_Future_Direction.pdf`.

[20] Infinera, editor. *Packet-optical. The Infinera way.* Accesible en `https://www.infinera.com/wp-content/uploads/infinera-ebook-packet-optical.pdf`.

[21] Amoldeep Singh, Kapal Dev, Harun Siljak, Hem Dutt Joshi, and Maurizio Magarini. Quantum internet- applications, functionalities, enabling technologies, challenges, and research directions. 2021. arXiv: 2101.04427.

[22] Eric and Chitambar. Quantum resource theories. *Reviews of Modern Physics*, 91(2), Apr 2019. American Physical Society.

[23] M Peev et al. The SECOQC quantum key distribution network in vienna. *New Journal of Physics*, 11(7):075001, jul 2009. IOP Publishing.

[24] M. Sasaki et al. Field test of quantum key distribution in the tokyo qkd network. *Opt. Express*, 19(11):10387–10409, May 2011. OSA.

[25] Yu-Ao Chen and et. al. An integrated space-to-ground quantum communication network over 4,600 kilometres. *Nature*, 589, 01 2021.

[26] V. Martin et al. Quantum aware sdn nodes in the madrid quantum network. In *2019 21st International Conference on Transparent Optical Networks (ICTON)*, pages 1–4, 2019.

[27] Shi-Jie Wei, Tao Xin, and Gui-Lu Long. Efficient universal quantum channel simulation in ibm's cloud quantum computer, 2017.

[28] Sowmitra Das, Md. Saifur Rahman, and Mahbubul Alam Majumdar. Design of a quantum-repeater using quantum-circuits and benchmarking its performance on an ibm quantum-computer, 2020.

[29] Fermín Galán Márquez. On scenario-based model-driven configuration management for flexible networking experimentation infrastructures. Abril 2010.

[30] Bikash K. Behera, Tasnum Reza, Angad Gupta, and Prasanta K. Panigrahi. Designing quantum router in ibm quantum computer. *Quantum Information Processing*, 18(11), Sep 2019. Springer Science and Business Media LLC.

[31] Diego Albuquerque, Bruno Cafeo, Alessandro Garcia, Simone Barbosa, Silvia Abrahão, and António Ribeiro. Quantifying usability of domain-specific languages: An empirical study on software maintenance. *Journal of Systems and Software*, 101:245–259, 03 2015.

[32] Fausto Giunchiglia and Toby Walsh. A theory of abstraction. *Artificial Intelligence*, 57(2):323–389, 1992.

[33] R. Taylor, N. Medvidovic, and E. Dashofy. *Software architecture.* John Wiley & Sons, 2010. ISBN 9780470167748.